NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AUTOMATED LOGISTICS PLANNING
USING HISTORICAL ANALOGIES

by

Mark J. Davis

June 1987

Thesis Advisor: Neil C. Rowe

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution is unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) Code 52	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
AUTOMATED LOGISTICS PLANNING USING HISTORICAL ANALOGIES (u)

12 PERSONAL AUTHOR(S)
Davis, Mark J.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year Month Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1987 June	113

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Automated Logistics Planning Systems; Analogies; Tactical Logistics Estimates

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The current method for creating tactical logistics estimates in the Army inadequately incorporates historical data on the actual consumption of supplies. The automated-logistics-planning system described in this thesis addresses this deficiency. The program developed in this research produces general estimates for selected supply items by referencing equations and variables from current Army planning documents and performing the necessary calculations. The program uses reasoning to identify previous operations which are analogous to the current operation. Separate criteria are used to identify the strongest analogies to the current operation for each of five categories of supply items. Information contained in the historical records of the three strongest analogies in each category is used to revise the general estimates. The revised estimates are hopefully more accurate in predicting actual supply requirements for the current operation than

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. Neil C. Rowe	22b TELEPHONE (Include Area Code) (408) 646-2462 22c OFFICE SYMBOL 52Rp

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

#19 ABSTRACT (Continued)

the estimates generated by formula alone.

Approved for public release; distribution is unlimited.

Automated Logistics Planning Using Historical Analogies

by

Mark J. Davis
Captain, United States Army
B.S., United States Military Academy, 1980

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1987

ABSTRACT

The current method for creating tactical logistics estimates in the Army inadequately incorporates historical data on the actual consumption of supplies. The automated-logistics-planning system described in this thesis addresses this deficiency. The program developed in this research produces general estimates for selected supply items by referencing equations and variables from current Army planning documents and performing the necessary calculations. The program uses reasoning to identify previous operations which are analogous to the current operation. Separate criteria are used to identify the strongest analogies to the current operation for each of five categories of supply items. Information contained in the historical records of the three strongest analogies in each category is used to revise the general estimates. The revised estimates are hopefully more accurate in predicting actual supply requirements for the

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. BACKGROUND

The conduct of military operations is inextricably interwoven with the ability to support them. One of the five Principles of War, Mass, has as its premise that superior combat power must be concentrated at the decisive place and time in order to achieve decisive results. Logistics planning is fundamental to achieving this aim. It defines realistic scenarios for the study of alternative courses of action, and determines directly the length of time that weapons and units can be effective.

In the United States Army, the tactical commander is responsible for logistics planning. He normally has assigned to him certain staff who assist him in estimating the logistics requirements of anticipated or considered actions, and in incorporating whatever limitations exist into his battle planning. Logistics planning at this level is dominated by the creation of estimates and the reporting of equipment, supply, and personnel status. Logistics estimates are created often and represent a significant work load for those soldiers whose job it is to produce them for the commander.

There are currently no automated methods for assisting the tactical logistics planner in creating such estimates. Training Management and Control System (TMACS) is a software system currently in use in the Army which assists operational planners in programming and budgeting time, money, and supplies for training exercises [Ref. 1]. It does not, however, satisfy the need for an automated aid in preparing logistics estimates for actual tactical operations and in the conduct of contingency planning.

At the same time, the military of the Soviet Union places great emphasis on automated means for enhancing their theory of control. The use of computers to perform referencing and calculations is actively pursued. Specifically, the referencing of applicable planning factors and execution of mathematical formulae in combat planning is an area in which they have exhibited considerable interest [Ref. 2].

This thesis investigates the nature of logistics estimation in the tactical environment. It identifies the merits of employing an automated system to perform some of the current labor-intensive manual referencing of equations and planning factors involved in creating a logistics estimate. A prototype of such an automated

system is one of the products of this research. Execution of the prototype on a sample database produces output that is easily understood by the logistics planner, and offers significant time savings in the preparation of logistics estimates.

Reasoning is used in the prototype system in an attempt to provide more accurate logistics estimates than are currently provided by strict adherence to the procedures outlined in current Army planning documents. The reasoning algorithm is the most interesting facet of the prototype. The aim of all of this work is to provide the logistics planner with an aid in providing timely, useable, and well considered logistics estimates to the battlefield commander.

B. RESEARCH TOPICS

The first area of research centers around the process by which logistics estimates are created today. This process is strictly manual. A single reference document, Army field manual FM 101-10-1, contains equations and planning factors which the logistics planner uses in calculating the quantity of several supply items required to conduct an operation [Ref. 3]. The equations and planning factors used in these calculations are dependent upon several key descriptions about the operation for which the estimate is being prepared. The following tasks need to be performed by an automated system to duplicate the actions of the human logistics planner:

1. identification of which attributes of an operation are critical to the selection of appropriate equations and planning factors needed to calculate logistics estimates
2. determination of how such information can best be obtained from the user of the prototype program
3. performance of the aforementioned references and calculations.

The second area of research is concerned with reasoning about the similarity between operations. Operations which have the same value for certain key operation attributes can be considered analogous to the current operation. These analogies are then evaluated to determine the strength of the similarity between them and the current operation. The criteria used to establish the analogous nature of previous operations and the criteria used to establish the strength of the similarity between operations are both defined in the program, but can easily be modified to reflect user guidelines. Once the strongest analogies to the current operation have been determined, research into how the data contained in the historical records of these operations can be used to adjust the previously calculated estimates can be pursued.

The third area of research involves determining the method by which adjustments to the original logistics estimates are made. Analogous historical records contain information on both the original estimate and the actual consumption for each of the items of supply for which the program creates an estimate. The error percentage for each of the three strongest analogies in estimating the actual consumption of each of the supply items is calculated. Originally, an equal weighting of the error percentages calculated for the three strongest analogies was used to adjust the estimates for the current operation. The final version of the prototype handles the weighting somewhat differently. Specifically, the error percentages are weighted according to the strength of the similarity between the operation which generated them and the current operation. The composite error percentage is then used in a calculation which yields an adjustment to the original estimate.

The fourth area of research deals with the format and structure of program input and output. The decision was made to utilize menus as much as possible for input from the user. Errors are common when using a program of this sort. Where menus are not practical, escape routines were included to allow recovery from input errors. Well structured, explanatory output is extremely valuable in understanding the behavior of the program. An effort was made to produce one page documents. The reasoning algorithm of the program is illustrated in tabular, one page summaries to assist the user in understanding how the reasoning is conducted.

C. THESIS ORGANIZATION

Chapter II discusses the manner in which tactical logistics planning is conducted today. The data elements needed to conduct such planning are identified and their relationship to the key operational planning document, the operation order (OPORDER), is explained. The linear equation model used to calculate logistics estimates is outlined, as well as the source and questionable validity of the planning factors. The difficulties faced by the tactical logistics planner in producing logistics estimates are identified. A short discussion on the type of reasoning expected of the logistics planner in creating estimates concludes the chapter.

Chapter III outlines an artificial-intelligence approach to creating the logistics estimate. An initial discussion identifies the anticipated benefits of using an automated aid in assisting the logistics planner in creating the estimate. A lengthy discussion of the reasoning done in the program follows. Specifically, the use of reasoning to identify the similarity between operations is described. Examples are given to illustrate

how reasoning is actually conducted in the program. The examples are helpful in understanding the two step approach to analogy evaluation and the selection of operation records for inclusion in the algorithm for adjusting estimates. The importance of simple input and output formats is highlighted, with emphasis on designing program output that reflects the reasoning which takes place in a program. High-level design decisions in development of the prototype are explained and program behavior is described in detail.

Chapter IV discusses the potential of the thesis for assisting the tactical logistics planner in creating logistics estimates. The program represents a new approach to creating these estimates by applying real-world experience in adjusting estimates. Limitations of the program are discussed, as well as possible enhancements. The concluding discussion of the chapter identifies how the application of an automated logistics planning system of this type could be used in other related problem domains.

The appendices are vital to understanding the workings of the program. Appendix A and Appendix B are demonstrations of the program execution of the logistics-estimate-creation module of the program. The demonstrations involve two very different operations. The logistics estimate created in Appendix A is for an operation conducted in Europe in a temperate environment. The logistics estimate created in Appendix B is for an operation conducted in Korea in a cold environment. Each demonstration includes the following items:

1. a sample interactive session in which the user inputs the numbers of weapons and major end items in the task force
2. a sample interactive session in which the user assign values to several attributes describing the operation
3. a one page document produced by the program listing all operations which meet the criteria for being considered analogous to the current operation
4. a series of one page documents illustrating the reasoning in the program determining the strongest analogies to the current program for each category of supply
5. a logistics estimate for the operation.

Appendix C contains sample program execution of the other modules of the program. There is a sample interactive session which updates the historical record of an operation with actual consumption data. There is a sample interactive session which deletes the historical record of an operation from the historical files maintained by the program. There is a sample output produced by selecting the print-directory

module of the program which lists the unit name, date, and update status for all operations in the historical files. Appendix C also contains a sample of the historical record for an operation produced by the print-history module of the program.

Appendix D is the program implemented in Pascal. Appendix E is a partial implementation of the program in Common Lisp.

II. PROBLEM DEFINITION

A. TACTICAL LOGISTICS PLANNING TODAY

In tactical units of the United States Army, the staff officer charged with the creation of logistics estimates on behalf of the commander is the G4/S4 officer. A typical logistics plan might include estimates for the following categories of supplies:

1. water
2. subsistence
3. fuel
4. ammunition
5. general supplies.

To calculate estimates for supply items in each of these categories, the G4/S4 uses equations contained in Army field manual FM 101-10-1. These equations can be thought of as rules. Certain attributes of an operation determine either individually or in combination the equation to be used in calculating specific estimates. These attributes are contained in a key operational planning document called the operation order (OPORDER). The G4/S4 must obtain a draft of the OPORDER or otherwise reference these data elements before a logistics plan can be prepared. Attributes which directly influence how an operation is conducted and the type and quantity of supplies consumed in its execution include the following:

1. mission to be performed
2. climate in which the operation is to be conducted
3. area of the world in which the operation is to be conducted
4. type, size, and personnel strength of the task force
5. expected intensity of combat
6. ration policy during the operation.

There are many other factors which impact on the conduct of an operation. Current Army planning documents, however, use only the six attributes identified above in selecting appropriate equations from FM 101-10-1. The G4/S4 manually references the aforementioned attributes of an operation and indexes both an equation and a planning factor for use in calculating specific supply estimates. These attributes are contained in the operation order (OPORDER) for the operation.

For example, to calculate the expenditure of 5.56mm rifle ammunition, the following steps are taken. First, the mission and anticipated combat intensity of the operation are identified. Second, the attributes are used to select the appropriate equation and planning factor from the section in FM 101-10-1 covering ammunition estimates. In this case, the general equation to estimate 5.56mm rifle ammunition is:

weapons * planning factor = estimate.

Maintaining data on the number of rifles in the task force which expend 5.56mm ammunition is another task which the logistics planner is charged to perform. The same methodology is used for the other estimates.

The equations used to create the supply estimates are simple and easy to understand. The equations make intuitive sense as they are a function of the number of rifles in the task force and a single planning factor. The planning factors themselves are another subject. There has been much debate regarding their validity. Much of the data in FM 101-10-1 was originally based upon experience in World War II and Korea. Changes to the data have been made to reflect more recent experience and the results of combat modelling and simulation, but distrust of the accuracy of the planning factors continues. [Ref. 4]

The current method of creating estimates has another more serious shortcoming. The planning factors used in the simple linear equation model described above yield, at best, very generalized estimates. The data does not explicitly account for variable factors such as visibility, terrain, and the availability of close air support. Each of these impact significantly on the conduct of combat operations and on the rate at which supplies are consumed. It is necessary, therefore, that the logistics planner apply reasoned judgement in adjusting the estimates to reflect the particular set of attributes of the current operation.

There is no standard policy or guideline for the logistics planner to follow in making these reasoned judgements. Every commander hopes to have an experienced logistics planner who can rely upon personal experience or insightful after-action-reports to provide the basis for adjusting the standard estimates. All too often, the commander is without such a key individual. In addition, the commander and the rest of his staff continually create contingency and alternative operation plans. These plans require that logistics planning be conducted with a corresponding cost in time and effort. The logistics planner is seemingly always late in delivering logistics estimates to the commander while attempting to produce complete, researched, and well prepared plans.

An additional concern of the tactical logistics planner is the recording and retrieval of data on the actual consumption of supplies. A complete accounting of the logistically significant data about an operation is valuable in preparing future logistics estimates. A logistics estimate of an operation, together with corresponding actual consumption data for that operation, support the kind of reasoning described earlier. The time and effort it takes to record actual consumption data and link it with its associated logistics estimate often frustrates intentions to create complete historical records. The result is that such historical records do not exist in many units. When new personnel arrive and assume responsibility for logistics planning in the unit, they are without the benefit of historical data on which to base their estimates.

In partial summary, the Army realizes that reliance upon the equations in FM 101-10-1 will not yield acceptable estimates in all cases. The current method of computing estimates is simple to follow, but requires a considerable amount of the logistics planner's most precious commodity--time. There is also an acknowledged need for applying experience in logistics planning to the job of improving the accuracy of these general estimates. Experience is a hard thing to quantify, however, and many tactical logistics planners are not experienced. For these reasons, any system which significantly assists the logistics planner in creating estimates and performing associated tasks without imposing additional requirements would be of great value.

B. SURVEY OF PREVIOUS RELATED WORK

Commercially available spreadsheet programs perform the kind of referencing and calculations involved in creating logistics estimates as directed in current Army planning documents. Spreadsheets have been used in many business applications. They are able to adjust previous data to reflect changes in the values of program variables. The United States Army Logistics Center has developed several templates using a popular spreadsheet program, LOTUS 1-2-3. The templates use this program to create logistics estimates with equations and planning factors obtained from FM 101-10-1 and variables representing the personnel strength and the equipment composition of a task force. The templates are intended for use by logistics planners in creating estimates for Class I (subsistence) and Class III (POL) supply items. There are no templates currently developed to assist the logistics planner in creating logistics estimates for supply items in the other categories of supply.

The templates have two major limitations. First, each of the templates is written to run independently. This means that a logistics planner who desires to run both of these programs must run them separately. The estimates generated by the two programs must be abstracted onto a single document along with estimates for other supply items when creating an overall estimate for the operation being conducted. Second, the template used to create these logistics estimates assumes a static task force composition. The templates use the authorized numbers of personnel and equipment for the unit rather than the actual numbers taking part in the operation. The authorization document used for this purpose is the Modified Table of Organization and Equipment (MTOE) for the unit. The templates provide a valuable service to the logistics planner by automatically completing required paper work for requisitioning supplies. The inflexibility of the templates in accepting changes to the task force composition, however, causes them to fail to make full use of the power of the spreadsheet program, and limits the utility of the templates as an automated planning tool.

There has been little published on the use of analogies in creating logistics estimates. Much of the literature in operations research focuses on the use of numerical analysis in creating and revising estimates [Ref. 5]. Strictly numeric techniques, however, sometimes fail to model and predict physical phenomenon. Optimization techniques involving numerical analysis do not work when the result of the analysis is a guess or estimate of a future outcome [Ref. 6]. Such techniques are best suited to problems where the possible outcomes are known in advance. Logistics estimation at the tactical level is not an exact science and is resistant to attempts at applying such techniques. Non-numeric reasoning, called heuristic reasoning, has proven valuable in prediction and forecasting when numerical analysis has proven difficult or unacceptably costly. While the use of heuristics does not guarantee optimal solution, it can produce acceptable results.

Reasoning about the similarities between situations is an interesting subject in artificial intelligence research that has promise for assisting the prediction efforts of logisticians [Ref. 7]. Psychometric literature includes research describing how humans search for similarities between previous and new situations in an attempt to exploit knowledge about previous situations in order to improve current performance [Ref. 8]. These parallel academic research efforts support the research of this thesis in investigating the possible use of historical analogies to construct logistics estimates.

Some situations can be described by attributes or properties. It may be helpful to compare the values for certain attributes or properties of one situation with the values for the same attributes or properties of other situations in an attempt to establish the similarities between them. If similarities between situations exist, then it may be possible to infer some information about one such situation from information already available about the other. With regard to logistics estimates, the influence of certain operation attributes on the actual consumption of individual supply items might be able to be inferred from available data on the actual consumption of supplies of previously conducted operations. While the use of this technique to improve the accuracy of tactical logistics estimates has not been explicitly detailed in technical literature, it conceivably offers great potential as an estimation tool for the logistics planner.

III. PROGRAM DESIGN AND IMPLEMENTATION

A. MAJOR PROGRAM FUNCTIONS

The program was designed to assist the tactical logistics planner in performing many of the tasks associated with creating logistics estimates. To this end, there are five major functions performed by the program.

The first and most important function is to create a logistics estimate for an operation using information supplied by the user in an interactive session and with data retrieved from historical files maintained in secondary storage. This function is the real heart of the program. It references equations, performs calculations, and contains code that reasons about the similarity of operations. The output generated by this part of the program is of particular interest and is discussed later in the chapter.

The second function performed by the program is to update a record of a logistics estimate with data on the actual consumption resulting from the conduct of an operation. This function would be employed after an operation had already been conducted, and supplements or replaces much of the effort spent in preparing logistics after-action reports. The value of this function lies not only in its automation of the report generation task, but also in the storage of this information in the same data structure as the original logistics estimate for easy retrieval and logical representation.

The third function is to delete from the historical files the record of a previously created logistics estimate. Deletions of this sort are desireable when an operation for which a logistics estimate has been created is not conducted. If the actual consumption data in the record is significantly influenced by a factor which is considered an abnormal occurrence, the logistics planner may want to preclude its use in future program references by deleting it from the historical files. Another implementation might use a boolean flag for this purpose.

A fourth function is to print a directory with data on all the records of logistics estimates in the historical files. This is an important program function because it provides the user of the program with visibility over records currently in the historical files without using some of the more complex functions of the program. One of the data elements displayed in the directory output is whether or not each of the logistics estimates has been updated. This provides an easy method for the logistics planner to see whether an update needs to be made to a particular logistics estimate.

The last function of the program provides the capability of searching through the historical files which the program has previously created and prints a one page summary of all of the critical information associated with each of the operations. Such information includes not only the original logistics estimate, but also actual consumption data for each of the items of supplies for which an estimate was prepared. This data would be input using the record update function of the program previously described.

Figure 3.1 shows the top level design of the program. Each of the program functions described in this section is implemented as a module in the program.

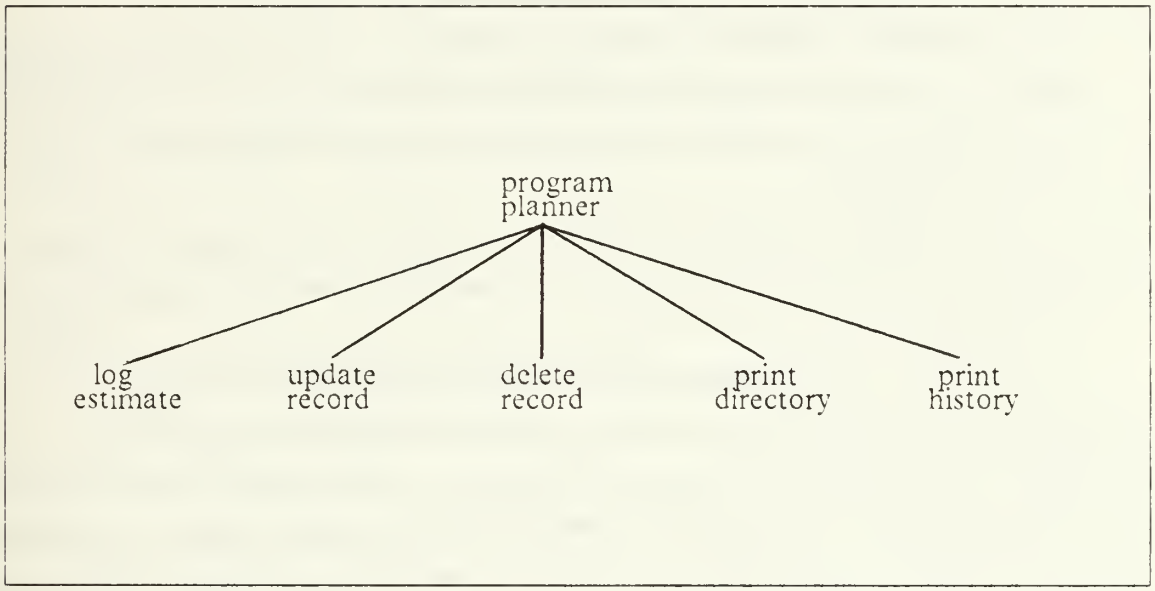

Figure 3.1 Program Top Level Design.

B. DATA STRUCTURES

The key design decision with regard to data structures was the manner in which all of the information regarding the creation of a logistics estimate would be stored. A record called an *oprecord* (operation record) was chosen. It contains fields which describe all of the scenario data used by the program to create a general estimate for each of the supply items for which the program creates an estimate. Additional scenario data is collected and stored for the purpose of reasoning about the similarities

between operations. The oprecord also contains a multi-dimensional array named *consumption* which contains information about each of the supply items for which the program creates an estimate. This information includes :

1. supply item name
2. general estimate for the supply item generated by performing the calculation of equations referenced from Fm 101-10-1 and using planning factors found in reference 4
3. adjustments to this estimate determined by the reasoning and adjustment techniques of the program
4. revised estimate obtained by adding the general estimate and the adjustment
5. actual consumption of the supply item during the conduct of the operation, if such data has been placed into the record.

Information about the composition of the task force conducting the operation is stored in an array called *task force*. The information contained in this data structure is not permanent and is lost after the logistics estimate function of the program has completed execution.

The choices for data structures in Pascal are satisfactory [Ref. 9]. The availability of user-defined file types made file input and output straightforward. After programming portions of the program in LISP as part of other course work, LISP appears to be at least as desirable a programming language for an application of this sort [Ref. 10]. In particular, LISP structures and flavors require fewer variable declarations, and results in program code that is easier to read. They also offer no temptation to rely on function side effects to assign values to fields in data structures. A partial implementation of the program in Common Lisp is included as Appendix E.

C. CREATING A LOGISTICS ESTIMATE

1. Input

The first input expected from the user after selecting the log-estimate module of the program is the composition of the task force conducting the operation. An example interactive session in which the user supplies the number of each kind of weapon and major end item is found in Appendix A. The selection of weapons and major end items included in the program was a design decision. A major criterion for inclusion was the availability of planning factor data in FM 101-10-1 and other widely used planning references [Ref. 4].

The program then proceeds to query the user about certain attributes describing the operation for which the program will create a logistics estimate.

Examples of the queries posed to the user are contained in the demonstrations in Appendix A and Appendix B. User responses to these queries are stored in the operation record (oprecord) identified earlier.

An important design decision in handling program input from the user was reliance on menus. The program is intended to serve as a prototype of an actual logistics planning tool. The use of menus reduces the potential for user input errors when the program is used in the field where unfamiliar users may enter erroneous information. In addition, menus support the use of enumerated types. Enumerated types were deemed important in promoting clarity when reading the program and studying its design. The cost of using menus is more extensive input procedure coding.

A special point is made to ensure that the program does not impose any additional data gathering requirements upon the logistics planner beyond those currently in effect. All of the data requested by the program of the user is contained in the operational planning document called the OPORDER (operation order) or its draft referred to in Chapter I. The logistics planner can answer all program queries using only those sources of information to which he/she has routine accesss. In event that the operation order itself is automated, user input of some of this data might not be required.

2. **Database Operations and General-estimate Calculation**

The equations and planning factors used in calculating the general logistics estimates are implemented as procedures with extensive parameter lists. Information about the composition of the task force and certain aspects of the operation scenario are used to select the correct equations and variable values. The result of these calculations is the general estimate, and is equal to the result obtained by referencing FM 101-10-1 and performing the mathematics manually. This portion of the program is a single-purpose database algorithm. The computer performs this operation much faster, and with greater reliability than can a human. The estimates created by these calculations are then stored in the operation record (oprecord) of the logistics estimate.

A restriction of the program implementation is the use of array indices to reference task force data. Knowledge of the data structure containing information on the composition of the task force is used in performing the database operations. The referencing of this data could be accomplished differently. A Line Item Number (LIN) is uniquely associated with every weapon and major end item in the Army inventory. Program implementation could be changed to use this attribute to reference task force

composition rather than relying on data structure knowledge. This would be especially useful in an environment where the type of weapons and major end items were not constant. It also supports the principle of information hiding [Ref. 11].

3. **Reasoning**

The apparently intelligent behavior of the program is the result of the reasoning it does about the similarity between the current operation and previous operations existing in the historical files maintained by the program. The program accomplishes this kind of reasoning by comparing the values for attributes in the description of each previous operation with the values for the same attributes in the current operation. The program uses a formula to determine the strength of the similarity between the two operations. All previous operations are evaluated in this manner. Data contained in the operation records of the three most similar previous operations will be used in an adjustment algorithm to refine the general estimates calculated earlier, yielding more accurate estimates.

The first phase of the reasoning process identifies all of the previous operations that are considered analogous to the current operation. For a previous operation to be considered analogous, it must meet the following criteria.

1. The previous operation has actual consumption data.
2. The previous operation and the current operation must have the same mission.
3. The previous operation and the current operation must take place in the same area.
4. The previous operation and the current operation must take place in the same climate.
5. The previous operation and the current operation must take place under the same chemical defense posture.
6. The previous operation and the current operation must have the same combat intensity.
7. The previous operation and the current operation were both first day engagements or succeeding day engagements.

All previous operations are analyzed and a list of analogous operations created. The program generates output showing all previous operations meeting this criteria and considered analogous to the current operation. Examples of this output are found in Appendix A and Appendix B under the heading, Analogy Reasoning.

The list of analogous operations is treated as a candidate list from which up to three operations will be selected as input into an adjustment algorithm. The action of

the adjustment algorithm is to reflect knowledge about the past consumption of supplies in adjusting the general estimates obtained through the database computations.

The second phase of reasoning determines which of the candidate operations will be selected for the purpose of adjusting estimates. The program performs this reasoning by evaluating certain attributes of the scenario descriptions of all of the candidate operations, and allocates analogy strength points to operations which have the same values for those attribute as does the current operation. The three candidate operations with the greatest analogy strength points are selected for use in the adjustment algorithm.

The sophistication of the second phase of reasoning does not end here. The supply items for which the program creates logistics estimates are grouped into five categories. The consumption rates for the supply items in each of the categories are assumed to be directly influenced by the same set of operation attributes as are the other supply items in their category. The program is able to reason about which candidate operations are the strongest analogies to the current operation in each of these categories independently. This is important since the consumption rates of supply items within different categories of supply are influenced to varying degrees by similar factors and by different conditions than are supply items in other categories. The program adjusts the estimates of supply items in each of the categories by using the strongest analogies to the current operation for those attributes influencing consumption of supply items in that category. In this way, the program makes the most effective use of the data available on previous operations. The five categories of supply considered by the program are :

1. water
2. subsistence
3. fuel
4. ammunition
5. general supplies.

The attributes of an operation which are used in determining the strongest analogies in each of these categories are identified in the program output. Factors influencing the consumption rates of supplies are not equally significant. The program applies weighting factors to each attribute and sums the value of all attributes in determining the total strength points for a particular operation. Appendix A and

Appendix B both contain program output which illustrates this reasoning. Those entries in which a "yes" is marked had a match between the attribute value in the previous operation and the attribute value in the current operation. The weighting given to each of the attributes is found in parenthesis under the attribute name.

It is important to note the adaptability of this form of reasoning. Any change in attributes used in performing the kind of reasoning contained in the program can be easily made. Changes to the weighting given to any of the attributes can be changed by modifying a single value in the code. The analyst or combat modeler can change the action of the program by modifying the identity or the weightings given to attributes of the operation to reflect more accurately the influence of operation attributes affecting the consumption of supplies.

4. **Adjustment Algorithm**

After the strongest analogies for each of the categories of supply items have been determined, an adjustment to the general estimate for each of the supply items computed. The adjustment algorithm works this way.

An error percentage is calculated for each of the analogies with respect to the particular supply item being considered. This is done by taking the difference between the actual consumption and the general estimate for the supply item and dividing it by the general estimate. Once this has been done for each of the analogous operations, these error percentages are weighted by the analogy strengths of the operations from which they were calculated, and averaged together. The resultant error percentage is then multiplied by the general estimate for the same supply item in the current logistics estimate. The result is stored as the adjustment to this general estimate. In computing the final estimate for the supply item, the adjustment is added to the general estimate. Adjustments can be be positive, negative, or zero.

There are several ways in which the error percentages of the three strongest analogies can be averaged. One way is to weight the error percentages equally. Another way is to place greater weight on the error percentages obtained from more recently conducted operations. The current weighting strategy places greater weight on the error percentages of the strongest analogies. Testing of the results of the program has not been done, but this weighting strategy is expected to yield more accurate adjustments than other strategies.

5. Output

The program output generated by this function of the program has already been partially described. The logistics estimate itself is a one page document found in both Appendix A and Appendix B. The top half of the document contains a summary of the attributes of the operation. Next, a brief summary of the previous analogous operations used in the adjustment algorithm is provided. At present, this information pertains only to the ammunition adjustments. Ammunition adjustments are considered to be of particular interest since the majority of the estimates which the program creates are for ammunition supply items. At the bottom of the document is the logistics estimate for the operation. The document is otherwise self-explanatory.

D. UPDATING RECORDS

This function of the program is critical to the reasoning performed in the creation of the logistics estimate. After an operation has been conducted, one of the tasks required of the tactical logistics planner is to collect data on all of the supplies consumed. Such information is passed up the chain of command to satisfy reporting requirements, and retained by the unit conducting the operation for future planning purposes. The record update function of the program automates this task, and stores the actual consumption data in the same data structure as the original logistics estimate for that operation for future reference. An example of an interactive session which queries the user for the actual consumption data and acknowledges the update of the historical record of that operation is found in Appendix C.

E. DELETING RECORDS

Not all of the operations for which logistics estimates are created will actually be conducted. In other cases, the actual consumption figures for an operation may be suspect or otherwise undesireable for use in adjusting future logistics estimates. The usefulness of retaining such records in secondary storage is questionable. The program allows the records of such operations to be deleted. An example of an interactive user session using this function of the program is found in Appendix C.

F. PRINTING A DIRECTORY

This function is useful for the reasons discussed earlier. The information it provides is all that is necessary to uniquely identify each of the operations, unless more than one operation with the same mission for the same unit on the same day is created.

An additional identifier would then be necessary. Aside from listing all of the operations residing in secondary storage, this function identifies whether each operation has been updated or not. If a record has been updated, then the source of the update information is provided. The sources of update information are factual data and estimates. An example of the output generated by the print directory function of the program is found in Appendix C.

G. PRINTING HISTORICAL RECORDS

This function of the program generates one page summaries of all of the information available on each of the operations for which it has created a logistics estimate. Such a summary is invaluable to a logistics planner. It represents a significant effort in researching historical files and presenting the contents of the files in a clear and understandable format. This function of the program could be modified to produce the historical record of a single specified operation, rather than the records of all operations in the historical files. An example of the historical record of an operation is found in Appendix C.

IV. CONCLUSIONS

A. PROGRAM STRENGTHS

The program takes a different approach to the creation of logistics estimates from the one currently used by Army logistics planners. It automates the references and calculations performed by logistics planners in constructing estimates using Army directed algorithms. The program performs these tasks with significant benefits to the logistics planner. The program is fast, reliable, and tireless. It automates many of the other administrative tasks performed by the logistics planner in updating the historical records of operations and frees the logistics planner to do other things.

In addition to its automation benefits, the program conducts analysis that was heretofore conducted only by experienced or enterprising logisticians. This analysis takes the form of reasoning about the similarity between operations, and the evaluation of existing historical records. This type of reasoning is essential to improving the accuracy of logistics estimates generated by current estimation techniques. The ability of the logistics planner to direct the reasoning of his automated aid in revising estimates is extremely powerful. It allows the logistics planner to instantly respond to changing conditions in operational planning, and assists the logistics planner in creating consistent, reasoned estimates even when he/she lacks the requisite personal experience to conduct such reasoning. Logistics planners outside the Army can also benefit from such a reasoning facility. Inventory managers and production planners spend considerable time creating and revising estimates. The reasoning contained in the program could be adapted to assist them.

B. PROGRAM LIMITATIONS

The program does not create estimates for all of the supply items with which the logistics planner would be concerned. Likewise, it does not accept information on all of the different types and models of weapons and major end items which are currently in use in the field. One reason for these obvious limitations is that the program was designed as a prototype. The program needed to demonstrate function, not completeness. A serious factor affecting future efforts in the development of automated logistics planning aids of this type, however, is storage. Principally, the concern is main memory availability. The template used to create the task force in the

program was arbitrarily chosen. It was certainly a very small subset of the total inventory of weapons and major end items in the Army inventory.

A complete database of all weapons and major end-items in the United States Army does exist, and is called the Army Master Data File (AMDF). A program using a database of this size would be most appropriate in a war gaming or simulation application where memory resources necessary to support such a database would not be a constraint. If the program is implemented as a microcomputer-based system in tactical units, then a subset of the AMDF or the use of a task force template like the one used in the prototype would be appropriate.

An improvement of the program might allow the user to specify a set of weapons and major end items from the AMDF as one of the program functions. The resulting task force template would continue to be used until changed by the user by selecting a task force template creation function in the program. The overriding concern is that the task force database and template be tailored to meet the particular needs of the logistics planner using the program and be supported by the memory resources available.

Another limitation of the program is the temporary existence of task force composition data. Ideally, this information would be stored permanently with other data pertinent to the historical record of an operation. It currently is not. The loss of this data prohibits future reference to the composition of the task force involved in the conduct of a particular logistics operation. This limits some of the analysis which can be performed about an operation. Information on the composition of a task force may warrant inclusion in the historical record.

Current program design involves reading all of the historical records into main memory from secondary storage at the beginning of the program and writing them back into secondary storage at the end of the of the user session. The historical files take up considerable space when the program has been used to create a large number of logistics estimates. A more sophisticated data retrieval technique is necessary to reduce this dependency on main memory. If the historical files of several different units are stored together or shared in some type of distributed system, the data retrieval issue becomes even more important.

The decision to use Pascal as the implementation language in the prototype was made for convenience. Pascal may not be the best choice. Experimentation with Common Lisp in implementing portions of the program required fewer variables and

potentially less storage space than Pascal. In addition, the use of user defined structures in Common Lisp appeared to make the program easier to read and understand.

A significant limitation of the program is its rigidity. In particular, the program does not allow the user to change the value of an attribute in the description of an operation after a record for that operation has been created and placed into the historical records. War is unpredictable by nature. There may be times when the value of certain attributes in the operation description will not be as planned. It is unrealistic to expect the user of the program to recreate an entire logistics estimate because of a single change to the operation description.

In a combat simulation or modelling application of this program, the ability to quickly and easily change the value of attributes in the operation description and recreate an estimate would be very important. Any anticipated sensitivity analysis using this program would require this capability.

The estimates for Class II, Class IV, Class VII, Class VIII, and Class IX supplies are clearly unacceptable for the purposes of the tactical logistics planner. While the functions of the program worked well in creating estimates which may be superior to those generated by human planners by calculation methods alone, it is the level of abstraction at which the estimates were made that is the problem. No one orders such supplies by the short ton. Each supply item is uniquely identified by a stock number and ordered individually. There must be a more concerted effort by Army logisticians to provide planning equations and factors for selected individual supply items in these categories. The present level of abstraction serves only the needs of the transportation manager concerned with bulk planning data and the logistician at Army level and above.

The reasoning performed by the program represents a new approach to logistics planning and has several potential applications which will be discussed later. There is the potential however, for becoming overly impressed with the reasoning techniques of the program and trying to reason about too many factors at once, or about factors whose influence on supply consumption is uncertain. Some factors may certainly impact on how a particular operation is conducted, but may not influence the consumption of supplies in any consistent and meaningful way.

C. PROMISE FOR THE FUTURE

The program of this thesis could evolve into a valuable tool for the tactical logistics planner. Future work needs to be done to validate the approach of this research. The program offers many potential benefits to the logistics planner. It is fast and reliable. Much of the time consuming work now being done by humans can be confidently shifted to a computer. References and calculations are routine, monotonous, and unexciting aspects of the logistics estimation process. Such tasks are often performed poorly or in an untimely manner, especially in a high stress environment such as can be expected during combat operations. The program assists the logistics planner in the reasoning process as well. For the inexperienced planner, the built-in reasoning of the program may provide estimate revisions where they might not otherwise be possible. The experienced logistics planner can structure the reasoning of the program to reflect more accurately the influence of different operation attributes on the consumption rate of supplies. Even at the tactical level, sensitivity analysis can be performed by altering the values of various operation attributes and creating a new estimate for the revised situation. A final benefit of the program is the simplicity and speed with which consumption data is recorded. With the aid of the program, this all-too-often-neglected task may be routinely performed.

The ideas inherent to the creation of an automated logistics planning program can be used to design computer aided instruction programs for teaching logistics planners how to create logistics estimates. An explanation facility can be added to instruct the student as to which refences are made in preparing each of the individual supply item estimates in accordance with FM 101-10-1. The same instruction program could check student estimates against its own calculations for test operations and provide a general equation solutions when the student fails to provide a correct response.

A computer-aided instruction program might also be used to teach the basics of reasoning in adjusting logistics estimates. A sample session such as the one detailed in the appendices of this thesis might provide physical evidence of the influence that certain operation attributes may have on the consumption of supplies. Coupled with classroom instruction, such an approach might provide some needed experience to junior logisticians.

Logistics has been poorly integrated into most Army tactical simulations and war gaming exercises. Operational planning in these exercises fails to include realistic

consideration of logistics requirements. The reason cited most often in explaining this deficiency is that logistics estimates are too time consuming to create and slow the pace of the training. Use of an automatedlogistics-planning system like this one might be able to alleviate this problem to some degree. The program of this thesis clearly demonstrates that systems can be designed to create logistics estimates in a timely and responsive manner, and integrated into operational planning. It might be possible to expand these type of exercises in the future to include the participation of logistics planners in a meaningful way.

APPENDIX A
LOGISTICS ESTIMATE DEMONSTRATION #1

TASK FORCE INPUT

```
You will now begin building the task force.

enter the number of M2 INF FIGHTING VEH in your task force.
              40
enter the number of M3 CAV FIGHTING VEH in your task force.
               5
enter the number of M113 PERS CARRIER   in your task force.
              19
enter the number of M901 CBT VEH ITV    in your task force.
               5
enter the number of M125A1 81MM CARR    in your task force.
               9
enter the number of M106A1 107MM  CARR  in your task force.
              12
enter the number of M102 105MM HOW      in your task force.
               9
enter the number of M109 155MM SP HOW   in your task force.
               9
enter the number of M110 8in SP HOW     in your task force.
               9
enter the number of LAUN-LOAD MLRS      in your task force.
               0
enter the number of M163 VULCAN AIR DEF in your task force.
               3
enter the number of M730 CHAP AIR DEF   in your task force.
               3
enter the number of M1 TANK 105MM       in your task force.
              54
enter the number of M60 TANK 105MM      in your task force.
              54
enter the number of TOW LAUNCHER        in your task force.
              26
enter the number of M222 DRAGON LNCHR   in your task force.
              44
enter the number of M2 50 CAL MG        in your task force.
             123
enter the number of M60 MG              in your task force.
              49
enter the number of M16A1 RIFLE         in your task force.
            3000
```

```
The following questions describe the operation for which
the program will create a logistics requirements estimate.
All questions must be answered as directed.


Enter the date on which the operation is to commence.
Use the form dd/mm/yy

The date of the operation is 24/05/87

Is this the correct date?
Enter the number corresponding to your answer.
     1 - yes, date is correct
     2 - no,  date is incorrect


Enter the name of the unit for which this estimate or
update is being prepared. For example- 1/33rd

The name of the unit is 2/77th

Is this the correct unit name?
Enter the number corresponding to your answer.
     1 - yes, unit name is correct
     2 - no,  unit name is incorrect


Enter the number corresponding to the correct tf type.
     1 - armor
     2 - mechanized
     3 - infantry


Enter the number corresponding to the correct tf size.
     1 - battalion
     2 - brigade


Enter the number corresponding to the correct mission.
     1 - attack
     2 - defend


Is this the first day of this mission or is this a
succeeding day of a continuing mission.

Enter the correct number for your response
    1 - first day
    2 - succeeding day
```

Enter the name of the operation of which this mission is a part. For example- D-DAY

The name of the operation is Reforger

Is this correct ?
Enter the number corresponding to your answer.
1 - yes, operation name is correct
2 - no, operation name is incorrect

Enter the number corresponding to the correct area.
1 - conus
2 - europe
3 - korea

Enter the name of the country in which this mission will be conducted. For example- West Germany.
Be sure to capitalize the first letter in each word

The name of the country is West Germany

Is this correct ?
Enter the number corresponding to your answer.
1 - yes, country name is correct
2 - no, country name is incorrect

Enter the number corresponding to the correct climate.
1 - hot
2 - temperate
3 - cold

Enter the number corresponding to the correct intensity
1 - high
2 - mid
3 - low

Do you expect the task force to be in MOPP level three or MOPP level four during this mission.

Enter the correct number for your response
1 - yes
2 - no

Enter the number corresponding to the correct terrain
1 - open
2 - woods
3 - built up
4 - mountainous

Enter the number corresponding to the visibility
1 - good
2 - fair
3 - poor

Do you plan on significant Air Force ground support?
Enter the correct number for your response
    1 - yes
    2 - no

Enter the total number of personnel in the task force.

Enter the number corresponding to the ration policy
during the duration of this operation.

    1 - b_c_b
    2 - c_c_b

All of the available data on past operations has been evaluated to identify analogies to the current operation.

A previous operation is considered analogous to the current operation if the following conditions are satisfied:

1. The historical record of the previous operation has been updated with actual consumption data.
2. Both operations have the same mission.
3. Both operations took place in the same area of the world.
4. Both operations took place in the same climate.
5. Both operations took place under the same chemical defense mission oriented protective posture.
6. Both operations involved the same combat intensity.
7. Both operations were first day engagements or succeeding day engagements of the same mission type.

The following operations are analogous under this definition.

DATE	UNIT	MISSION	AREA	CLIMATE	MOPP	INTENSITY	FIRST/SUCCEEDING DAY
01/04/86	2/77th	ATTACK	EUROPE	TEMPERATE	YES	HIGH	FIRST DAY
04/04/86	2/77th	ATTACK	EUROPE	TEMPERATE	YES	HIGH	FIRST DAY
06/04/86	2/77th	ATTACK	EUROPE	TEMPERATE	YES	HIGH	FIRST DAY
10/05/86	3/24th	ATTACK	EUROPE	TEMPERATE	YES	HIGH	FIRST DAY
15/05/86	1/81st	ATTACK	EUROPE	TEMPERATE	YES	HIGH	FIRST DAY
03/04/87	2/77th	ATTACK	EUROPE	TEMPERATE	YES	HIGH	FIRST DAY
07/04/87	2/77th	ATTACK	EUROPE	TEMPERATE	YES	HIGH	FIRST DAY
22/04/87	2/77th	ATTACK	EUROPE	TEMPERATE	YES	HIGH	FIRST DAY
05/05/87	3/24th	ATTACK	EUROPE	TEMPERATE	YES	HIGH	FIRST DAY
16/05/87	2/77th	ATTACK	EUROPE	TEMPERATE	YES	HIGH	FIRST DAY

The analogous operations are evaluated on the strength of their similarity to the current operation in those areas pertinent to water supply consumption. Each of the points of similarity are weighted independently.

The weighting of each item is in parenthesis below the item name. i.e. (3) = 3 points for AF_GROUND_SUPPORT

Up to three previous operations are considered in the adjustment algorithm, with those operations with the highest number of quality points being chosen for this purpose.

DATE	UNIT	COUNTRY NAME	UPDATE SOURCE
		(2)	(1)
16/05/87	2/77th	YES	YES
05/05/87	3/24th	NO	NO
22/04/87	2/77th	NO	YES
07/04/87	2/77th	YES	NO
03/04/87	2/77th	YES	NO
15/05/86	1/81st	NO	YES
10/05/86	3/24th	YES	NO
06/04/86	2/77th	NO	NO
04/04/86	2/77th	NO	YES
01/04/86	2/77th	YES	NO

The analogous operations are evaluated on the strength of their similarity to the current operation in those areas pertinent to subsistence consumption. Each of the points of similarity are weighted independently.

The weighting of each item is in parenthesis below the item name. i.e. (3) = 3 points for AF_GROUND_SUPPORT

Up to three previous operations are considered in the adjustment algorithm, with those operations with the highest number of quality points being chosen for this purpose.

DATE	UNIT	UNIT NAME	UPDATE SOURCE
		(1)	(1)
16/05/87	2/77th	YES	YES
05/05/87	3/24th	NO	NO
22/04/87	2/77th	YES	YES
07/04/87	2/77th	YES	NO
03/04/87	2/77th	YES	NO
15/05/86	1/81st	NO	YES
10/05/86	3/24th	NO	NO
06/04/86	2/77th	YES	NO
04/04/86	2/77th	YES	YES
01/04/86	2/77th	YES	NO

The analogous operations are evaluated on the strength of their similarity to the current operation in those areas pertinent to general supply consumption. Each of the points of similarity are weighted independently.

The weighting of each item is in parenthesis below the item name. i.e. (2) = 2 points for AF_GROUND_SUPPORT

Up to three previous operations are considered in the adjustment algorithm, with those operations with the highest number of quality points being chosen for this purpose.

DATE	UNIT	AF GRND_SPT	TERRAIN	VISIBILITY	UNIT NAME	COUNTRY NAME	OPERATION NAME	UPDATE SOURCE
		(2)	(2)	(1)	(1)	(1)	(1)	(2)
16/05/87	2/77th	YES	NO	NO	YES	YES	NO	YES
05/05/87	3/24th	NO	NO	YES	NO	NO	NO	NO
22/04/87	2/77th	YES	YES	YES	YES	NO	YES	YES
07/04/87	2/77th	YES	NO	NO	YES	YES	YES	NO
03/04/87	2/77th	NO	NO	NO	YES	YES	YES	NO
15/05/86	1/81st	YES	YES	YES	NO	NO	NO	YES
10/05/86	3/24th	YES	NO	YES	NO	YES	NO	NO
06/04/86	2/77th	NO	NO	NO	YES	NO	YES	NO
04/04/86	2/77th	YES	YES	YES	YES	NO	YES	YES
01/04/86	2/77th	YES	NO	NO	YES	YES	YES	NO

The analogous operations are evaluated on the strength of their similarity to the current operation in those areas pertinent to fuel supply consumption. Each of the points of similarity are weighted independently.

The weighting of each item is in parenthesis below the item name. i.e. (2) = 2 points for AF_GROUND_SUPPORT

Up to three previous operations are considered in the adjustment algorithm, with those operations with the highest number of quality points being chosen for this purpose.

DATE	UNIT	AF GRND_SPT	TERRAIN	UNIT NAME	COUNTRY NAME	OPERATION NAME	UPDATE SOURCE
		(2)	(3)	(1)	(1)	(1)	(2)
16/05/87	2/77th	YES	NO	YES	YES	NO	YES
05/05/87	3/24th	NO	NO	NO	NO	NO	NO
22/04/87	2/77th	YES	YES	YES	NO	YES	YES
07/04/87	2/77th	YES	NO	YES	YES	YES	NO
03/04/87	2/77th	NO	NO	YES	YES	YES	NO
15/05/86	1/81st	YES	YES	NO	NO	NO	YES
10/05/86	3/24th	YES	NO	NO	YES	NO	NO
06/04/86	2/77th	NO	NO	YES	NO	YES	NO
04/04/86	2/77th	YES	YES	YES	NO	YES	YES
01/04/86	2/77th	YES	NO	YES	YES	YES	NO

The analogous operations are evaluated on the strength of their similarity to the current operation in those areas pertinent to ammo supply consumption. Each of the points of similarity are weighted independently.

The weighting of each item is in parenthesis below the item name. i.e. (3) = 3 points for AF_GROUND_SUPPORT

The three previous operations with the highest number of quality points are used in the adjustment algorithm.

DATE	UNIT	AF GRND_SPT	TERRAIN	VISIBILITY	UNIT NAME	COUNTRY NAME	OPERATION NAME	UPDATE SOURCE
		(3)	(2)	(1)	(1)	(1)	(1)	(1)
16/05/87	2/77th	YES	NO	NO	YES	YES	NO	YES
05/05/87	3/24th	NO	NO	YES	NO	NO	NO	NO
22/04/87	2/77th	YES	YES	YES	YES	NO	YES	YES
07/04/87	2/77th	YES	NO	NO	YES	YES	YES	NO
03/04/87	2/77th	NO	NO	NO	YES	YES	YES	NO
15/05/86	1/81st	YES	YES	YES	NO	NO	NO	YES
10/05/86	3/24th	YES	NO	YES	NO	YES	NO	NO
06/04/86	2/77th	NO	NO	NO	YES	NO	YES	NO
04/04/86	2/77th	YES	YES	YES	YES	NO	YES	YES
01/04/86	2/77th	YES	NO	NO	YES	YES	YES	NO

The task force has been built. Task force composition is

M2 INF FIGHTING VEH	40
M3 CAV FIGHTING VEH	5
M113 PERS CARRIER	19
M901 CBT VEH ITV	5
M125A1 81MM CARR	9
M106A1 107MM CARR	12
M102 105MM HOW	9
M109 155MM SP HOW	9
M110 8in SP HOW	9
LAUN-LOAD MLRS	0
M163 VULCAN AIR DEF	3
M730 CHAP AIR DEF	3
M1 TANK 105MM	54
M60 TANK 105MM	54
TOW LAUNCHER	26
M222 DRAGON LNCHR	44
M2 50 CAL MG	123
M60 MG	49
M16A1 RIFLE	3000

AUTOMATED LOGISTICS PLAN

DATE	24/05/87
UNIT	2/77th
TASK FORCE TYPE	ARMOR
TASK FORCE SIZE	BRIGADE
MISSION	ATTACK
DURATION	FIRST DAY
COMBAT INTENSITY	HIGH
OPERATION NAME	Reforger
AREA	EUROPE
COUNTRY	West Germany
CLIMATE	TEMPERATE
TERRAIN	OPEN
VISIBILTY	FAIR
AF GROUND SUPPORT	YES
MOPP LEVEL 3/4	YES
PERSONNEL STRENGTH	3500
RATION POLICY	c_c_b

HISTORICAL DATA AVAILABLE YES

DATE	22/04/87	04/04/86	16/05/87
UNIT	2/77th	2/77th	2/77th

LOGISTICS ESTIMATE

SUPPLY ITEM	GENERAL EST.	ADJUSTMENTS	FINAL EST.	
water	16170	2556	18726	gallons
B rations	3500	-527	2973	meals
MRE rations	7000	2564	9564	meals
class II supplies	6	1	7	STONS
diesel fuel	69530	8888	78418	gallons
class IV supplies	7	1	8	STONS
tank ammo 105mm	5616	-616	5000	rounds
TOW ammo	182	66	248	rounds
DRAGON ammo	88	-18	70	rounds
Howitzer ammo 105mm	3384	1137	4521	rounds
Howitzer ammo 155mm	3366	-355	3011	rounds
Howitzer ammo 8in	2592	520	3112	rounds
Vulcan ammo 20mm	11952	4117	16069	rounds
Mortar ammo 81mm	873	-163	710	rounds
Mortar ammo 107mm	1308	300	1608	rounds
MG ammo .50 caliber	21525	-5600	15925	rounds
MG ammo 7.62mm	21217	-6046	15171	rounds
rifle ammo 5.56mm	297000	3470	300470	rounds
class VII supplies	26	4	30	STONS
class VIII supplies	2	-1	1	STONS
class IX supplies	4	4	8	STONS

APPENDIX B
LOGISTICS ESTIMATE DEMONSTRATION #2

TASK FORCE INPUT

```
You will now begin building the task force.

enter the number of M2 INF FIGHTING VEH in your task force.
              40
enter the number of M3 CAV FIGHTING VEH in your task force.
               5
enter the number of M113 PERS CARRIER   in your task force.
              19
enter the number of M901 CBT VEH ITV    in your task force.
               5
enter the number of M125A1 81MM CARR    in your task force.
               9
enter the number of M106A1 107MM  CARR  in your task force.
              12
enter the number of M102 105MM HOW      in your task force.
               9
enter the number of M109 155MM SP HOW   in your task force.
               9
enter the number of M110 8in SP HOW     in your task force.
               9
enter the number of LAUN-LOAD MLRS      in your task force.
               0
enter the number of M163 VULCAN AIR DEF in your task force.
               5
enter the number of M730 CHAP AIR DEF   in your task force.
               3
enter the number of M1 TANK 105MM       in your task force.
              54
enter the number of M60 TANK 105MM      in your task force.
              54
enter the number of TOW LAUNCHER        in your task force.
              26
enter the number of M222 DRAGON LNCHR   in your task force.
              44
enter the number of M2 50 CAL MG        in your task force.
             123
enter the number of M60 MG              in your task force.
              49
enter the number of M16A1 RIFLE         in your task force.
            3000
```

```
The following questions describe the operation for which
the program will create a logistics requirements estimate.
All questions must be answered as directed.


Enter the date on which the operation is to commence.
Use the form dd/mm/yy

The date of the operation is 03/03/87

Is this the correct date?
Enter the number corresponding to your answer.
     1 - yes, date is correct
     2 - no,  date is incorrect


Enter the name of the unit for which this estimate or
update is being prepared. For example- 1/33rd

The name of the unit is 1/11th


Is this the correct unit name?
Enter the number corresponding to your answer.
     1 - yes, unit name is correct
     2 - no,  unit name is incorrect


Enter the number corresponding to the correct tf type.
     1 - armor
     2 - mechanized
     3 - infantry


Enter the number corresponding to the correct tf size.
     1 - battalion
     2 - brigade


Enter the number corresponding to the correct mission.
     1 - attack
     2 - defend


Is this the first day of this mission or is this a
succeeding day of a continuing mission.

Enter the correct number for your response
    1 - first day
    2 - succeeding day
```

Enter the name of the operation of which this mission is a part. For example- D-DAY

The name of the operation is Rising Star

Is this correct ?
Enter the number corresponding to your answer.
1 - yes, operation name is correct
2 - no, operation name is incorrect

Enter the number corresponding to the correct area.
1 - conus
2 - europe
3 - korea

Enter the name of the country in which this mission will be conducted. For example- West Germany.
Be sure to capitalize the first letter in each word

The name of the country is Korea

Is this correct ?
Enter the number corresponding to your answer.
1 - yes, country name is correct
2 - no, country name is incorrect

Enter the number corresponding to the correct climate.
1 - hot
2 - temperate
3 - cold

Enter the number corresponding to the correct intensity
1 - high
2 - mid
3 - low

Do you expect the task force to be in MOPP level three or MOPP level four during this mission.

Enter the correct number for your response
1 - yes
2 - no

Enter the number corresponding to the correct terrain
1 - open
2 - woods
3 - built up
4 - mountainous

Enter the number corresponding to the visibility
1 - good
2 - fair
3 - poor

Do you plan on significant Air Force ground support?
Enter the correct number for your response
    1 - yes
    2 - no

Enter the total number of personnel in the task force.

Enter the number corresponding to the ration policy
during the duration of this operation.

    1 - b_c_b
    2 - c_c_b

All of the available data on past operations has been evaluated to identify analogies to the current operation.

A previous operation is considered analogous to the current operation if the following conditions are satisfied:

1. The historical record of the previous operation has been updated with actual consumption data.
2. Both operations have the same mission.
3. Both operations took place in the same area of the world.
4. Both operations took place in the same climate.
5. Both operations took place under the same chemical defense mission oriented protective posture.
6. Both operations involved the same combat intensity.
7. Both operations were first day engagements or succeeding day engagements of the same mission type.

The following operations are analogous under this definition.

DATE	UNIT	MISSION	AREA	CLIMATE	MOPP	INTENSITY	FIRST/SUCCEEDING DAY
01/02/86	1/11th	DEFEND	KOREA	COLD	YES	MID	SUCCEEDING DAY
22/02/86	1/11th	DEFEND	KOREA	COLD	YES	MID	SUCCEEDING DAY
04/03/86	1/11th	DEFEND	KOREA	COLD	YES	MID	SUCCEEDING DAY
12/01/87	2/22nd	DEFEND	KOREA	COLD	YES	MID	SUCCEEDING DAY
02/02/87	3/33rd	DEFEND	KOREA	COLD	YES	MID	SUCCEEDING DAY
23/02/87	2/22nd	DEFEND	KOREA	COLD	YES	MID	SUCCEEDING DAY

The analogous operations are evaluated on the strength of their similarity to the current operation in those areas pertinent to water supply consumption. Each of the points of similarity are weighted independently.

The weighting of each item is in parenthesis below the item name. i.e. (3) = 3 points for AF_GROUND_SUPPORT

Up to three previous operations are considered in the adjustment algorithm, with those operations with the highest number of quality points being chosen for this purpose.

DATE	UNIT	COUNTRY NAME	UPDATE SOURCE
		(2)	(1)
23/02/87	2/22nd	YES	NO
02/02/87	3/33rd	YES	NO
12/01/87	2/22nd	YES	NO
04/03/86	1/11th	YES	YES
22/02/86	1/11th	YES	YES
01/02/86	1/11th	YES	YES

The analogous operations are evaluated on the strength of their similarity to the current operation in those areas pertinent to subsistence consumption. Each of the points of similarity are weighted independently.

The weighting of each item is in parenthesis below the item name. i.e. (3) = 3 points for AF_GROUND_SUPPORT

Up to three previous operations are considered in the adjustment algorithm, with those operations with the highest number of quality points being chosen for this purpose.
.bf tiny

DATE	UNIT	UNIT NAME	UPDATE SOURCE
		(1)	(1)
23/02/87	2/22nd	NO	NO
02/02/87	3/33rd	NO	NO
12/01/87	2/22nd	NO	NO
04/03/86	1/11th	YES	YES
22/02/86	1/11th	YES	YES
01/02/86	1/11th	YES	YES

The analogous operations are evaluated on the strength of their similarity to the current operation in those areas pertinent to general supply consumption. Each of the points of similarity are weighted independently.

The weighting of each item is in parenthesis below the item name. i.e. (2) = 2 points for AF_GROUND_SUPPORT

Up to three previous operations are considered in the adjustment algorithm, with those operations with the highest number of quality points being chosen for this purpose.

DATE	UNIT	AF GRND_SPT	TERRAIN	VISIBILITY	UNIT NAME	COUNTRY NAME	OPERATION NAME	UPDATE SOURCE
		(2)	(2)	(1)	(1)	(1)	(1)	(2)
23/02/87	2/22nd	NO	YES	YES	NO	YES	YES	NO
02/02/87	3/33rd	NO	NO	NO	NO	YES	NO	NO
12/01/87	2/22nd	NO	NO	NO	NO	YES	YES	NO
04/03/86	1/11th	YES	YES	YES	YES	YES	NO	YES
22/02/86	1/11th	YES	NO	NO	YES	YES	YES	YES
01/02/86	1/11th	YES	YES	YES	YES	YES	YES	YES

The analogous operations are evaluated on the strength of their similarity to the current operation in those areas pertinent to fuel supply consumption. Each of the points of similarity are weighted independently.

The weighting of each item is in parenthesis below the item name. i.e. (2) = 2 points for AF_GROUND_SUPPORT

Up to three previous operations are considered in the adjustment algorithm, with those operations with the highest number of quality points being chosen for this purpose.

DATE	UNIT	AF GRND_SPT	TERRAIN	UNIT NAME	COUNTRY NAME	OPERATION NAME	UPDATE SOURCE
		(2)	(3)	(1)	(1)	(1)	(2)
23/02/87	2/22nd	NO	YES	NO	YES	YES	NO
02/02/87	3/33rd	NO	NO	NO	YES	NO	NO
12/01/87	2/22nd	NO	NO	NO	YES	YES	NO
04/03/86	1/11th	YES	YES	YES	YES	NO	YES
22/02/86	1/11th	YES	NO	YES	YES	YES	YES
01/02/86	1/11th	YES	YES	YES	YES	YES	YES

The analogous operations are evaluated on the strength of their similarity to the current operation in those areas pertinent to ammo supply consumption. Each of the points of similarity are weighted independently.

The weighting of each item is in parenthesis below the item name. i.e. (3) = 3 points for AF_GROUND_SUPPORT

The three previous operations with the highest number of quality points are used in the adjustment algorithm.

DATE	UNIT	AF GRND_SPT	TERRAIN	VISIBILITY	UNIT NAME	COUNTRY NAME	OPERATION NAME	UPDATE SOURCE
		(3)	(2)	(1)	(1)	(1)	(1)	(1)
23/02/87	2/22nd	NO	YES	YES	NO	YES	YES	NO
02/02/87	3/33rd	NO	NO	NO	NO	YES	NO	NO
12/01/87	2/22nd	NO	NO	NO	NO	YES	YES	NO
04/03/86	1/11th	YES	YES	YES	YES	YES	NO	YES
22/02/86	1/11th	YES	NO	NO	YES	YES	YES	YES
01/02/86	1/11th	YES	YES	YES	YES	YES	YES	YES

M2 INF FIGHTING VEH	40
M3 CAV FIGHTING VEH	5
M113 PERS CARRIER	19
M901 CBT VEH ITV	5
M125A1 81MM CARR	9
M106A1 107MM CARR	12
M102 105MM HOW	9
M109 155MM SP HOW	9
M110 8in SP HOW	9
LAUN-LOAD MLRS	0
M163 VULCAN AIR DEF	5
M730 CHAP AIR DEF	3
M1 TANK 105MM	54
M60 TANK 105MM	54
TOW LAUNCHER	26
M222 DRAGON LNCHR	44
M2 50 CAL MG	123
M60 MG	49
M16A1 RIFLE	3000

AUTOMATED LOGISTICS PLAN

DATE	03/03/87
UNIT	1/11th
TASK FORCE TYPE	ARMOR
TASK FORCE SIZE	BRIGADE
MISSION	DEFEND
DURATION	SUCCEEDING DAY
COMBAT INTENSITY	MID
OPERATION NAME	Rising Star
AREA	KOREA
COUNTRY	Korea
CLIMATE	COLD
TERRAIN	OPEN
VISIBILTY	GOOD
AF GROUND SUPPORT	YES
MOPP LEVEL 3/4	YES
PERSONNEL STRENGTH	3500
RATION POLICY	c_c_b

HISTORICAL DATA AVAILABLE YES

DATE	01/02/86	04/03/86	22/02/86
UNIT	1/11th	1/11th	1/11th

LOGISTICS ESTIMATE

SUPPLY ITEM	GENERAL EST.	ADJUSTMENTS	FINAL EST.	
water	12320	2004	14324	gallons
B rations	3500	2187	5687	meals
MRE rations	7000	-2000	5000	meals
class II supplies	6	2	8	STONS
diesel fuel	56257	8990	65247	gallons
class IV supplies	7	-2	5	STONS
tank ammo 105mm	2268	332	2600	rounds
TOW ammo	182	18	200	rounds
DRAGON ammo	88	-8	80	rounds
Howitzer ammo 105mm	2736	64	2800	rounds
Howitzer ammo 155mm	2916	-166	2750	rounds
Howitzer ammo 8in	2115	185	2300	rounds
Vulcan ammo 20mm	8100	1900	10000	rounds
Mortar ammo 81mm	360	-60	300	rounds
Mortar ammo 107mm	540	110	650	rounds
MG ammo .50 caliber	8856	144	9000	rounds
MG ammo 7.62mm	8673	-673	8000	rounds
rifle ammo 5.56mm	120000	-106000	14000	rounds
class VII supplies	26	4	30	STONS
class VIII supplies	2	0	2	STONS
class IX supplies	4	2	6	STONS

APPENDIX C
SAMPLE OUTPUT FROM OTHER PROGRAM FUNCTIONS

UPDATING A RECORD

```
You will now be asked information about the operation.
for which you have actual consumption data.


Enter name of unit which conducted the operation
For example- 1/33rd
The name of the unit was 2/77th

Is this the correct unit name?
Enter the number corresponding to your answer.
     1 - yes, unit name is correct
     2 - no,  unit name is incorrect


Enter the date on which the operation took place.
Use the form dd/mm/yy
The date of the operation was 24/05/87
Is this the correct date?
Enter the number corresponding to your answer.
     1 - yes, date is correct
     2 - no,  date is incorrect


Enter the number corresponding to the correct mission.
     1 - attack
     2 - defend


What was the source of the information for this update
Enter the correct number for your response
    1 - estimate
    2 - factual information


Enter the actual consumption for each of the supply
items that follow. If no actual consumption figures
are available, enter 0 .


Enter the number of water                gallons
        20000
Enter the number of B rations            meals
         3100
Enter the number of MRE rations          meals
         8100
Enter the number of class II supplies    STONS
             7
Enter the number of diesel fuel          gallons
        77700
```

```
Enter the number of class IV supplies   STONS
              8
Enter the number of tank ammo 105mm     rounds
           5117
Enter the number of TOW ammo            rounds
            247
Enter the number of DRAGON ammo         rounds
             75
Enter the number of Howitzer ammo 105mm rounds
           4600
Enter the number of Howitzer ammo 155mm rounds
           3040
Enter the number of Howitzer ammo 8in   rounds
           2930
Enter the number of Vulcan ammo 20mm    rounds
          15650
Enter the number of Mortar ammo 81mm    rounds
            705
Enter the number of Mortar ammo 107mm   rounds
           1572
Enter the number of MG ammo .50 caliber rounds
           1635
Enter the number of MG ammo 7.62mm      rounds
          16000
Enter the number of rifle ammo 5.56mm   rounds
         300000
Enter the number of class VII supplies  STONS
             29
Enter the number of class VIII supplies STONS
              1
Enter the number of class IX supplies   STONS
              9


The record has been updated.

Enter c to continue
```

```
You will now be asked information about the operation
that you want deleted.


Enter name of unit which conducted the operation
For example- 1/33rd
The name of the unit was 2/77th

Is this the correct unit name?
Enter the number corresponding to your answer.
     1 - yes, unit name is correct
     2 - no,  unit name is incorrect


Enter the date on which the operation took place.
Use the form dd/mm/yy
The date of the operation was 24/05/87
Is this the correct date?
Enter the number corresponding to your answer.
     1 - yes, date is correct
     2 - no,  date is incorrect


Enter the number corresponding to the correct mission.
     1 - attack
     2 - defend


The record was found and deleted.


                         Enter c to continue
```

DIRECTORY

DATE	UNIT	MISSION	UPDATED
01/04/86	2/77th	ATTACK	YES
04/04/86	2/77th	ATTACK	YES
06/04/86	2/77th	ATTACK	YES
10/05/86	3/24th	ATTACK	YES
15/05/86	1/81st	ATTACK	YES
03/04/87	2/77th	ATTACK	YES
07/04/87	2/77th	ATTACK	YES
22/04/87	2/77th	ATTACK	YES
05/05/87	3/24th	ATTACK	YES
16/05/87	2/77th	ATTACK	YES
24/05/87	2/77th	ATTACK	YES

Enter c to continue

HISTORICAL RECORD

DATE	24/05/87
UNIT	2/77th
TASK FORCE TYPE	ARMOR
TASK FORCE SIZE	BRIGADE
MISSION	ATTACK
DURATION	FIRST DAY
COMBAT INTENSITY	HIGH
OPERATION NAME	Reforger
AREA	EUROPE
COUNTRY	West Germany
CLIMATE	TEMPERATE
TERRAIN	OPEN
VISIBILTY	FAIR
AF GROUND SUPPORT	YES
MOPP LEVEL 3/4	YES
PERSONNEL STRENGTH	3500
RATION POLICY	c_c_b
UPDATE SOURCE	FACTUAL

HISTORICAL DATA AVAILABLE YES

DATE	22/04/87	04/04/86	16/05/87
UNIT	2/77th	2/77th	2/77th

LOGISTICS ESTIMATE

SUPPLY ITEM	GENERAL EST.	ADJUSTMENTS	FINAL EST.	ACTUAL CONS.	
water	16170	2556	18726	20000	gallo
B rations	3500	-527	2973	3100	meals
MRE rations	7000	2564	9564	8100	meals
class II supplies	6	1	7	7	STONS
diesel fuel	69530	8888	78418	77700	gallo
class IV supplies	7	1	8	8	STONS
tank ammo 105mm	5616	-616	5000	5117	round
TOW ammo	182	66	248	247	round
DRAGON ammo	88	-18	70	75	round
Howitzer ammo 105mm	3384	1137	4521	4600	round
Howitzer ammo 155mm	3366	-355	3011	3040	round
Howitzer ammo 8in	2592	520	3112	2930	round
Vulcan ammo 20mm	11952	4117	16069	15650	round
Mortar ammo 81mm	873	-163	710	705	round
Mortar ammo 107mm	1308	300	1608	1572	round
MG ammo .50 caliber	21525	-5600	15925	16350	round
MG ammo 7.62mm	21217	-6046	15171	16000	round
rifle ammo 5.56mm	297000	3470	300470	300000	round
class VII supplies	26	4	30	29	STONS
class VIII supplies	2	-1	1	1	STONS
class IX supplies	4	4	8	9	STONS

APPENDIX D
PASCAL PROGRAM

```
(*$S30000*)
program thesis (input,output);
const
   datesize             =  8; (*the width of the date field            *)
   unitsize             = 10; (*the width of the unit name field       *)
   num_supply_items     = 21; (*number of items of supply for which the
                                 program generates logistics estimates  *)
   supply_item_namesize = 19; (*the width of the supply item name field*)
   unit_of_measure_size =  7; (*the width of the unit of measure field *)
   maxfiles             = 20; (*number of operations in history files  *)
   max_analogies        =  3; (*max number of analogies used to adjust
                                 general estimates                      *)
   operation_name_length= 13; (*width of operation name field          *)
   country_name_length  = 14; (*width of country name field            *)

type
   unit_of_measure_string = packed array (.1..unit_of_measure_size.) of
                                 char;
   supply_item_string     = packed array (.1..supply_item_namesize.) of
                                 char;
   data = record
        supply_item        : supply_item_string;
        general_estimate   : integer;
        adjustments        : integer;
        final_estimate     : integer;
        actual_consumption : integer;
        unit_of_measure    : unit_of_measure_string
   end; (*end record data*)
   consumearray = array (.1..num_supply_items.) of data;
   datestring      = packed array (.1..datesize.) of char;
   unitstring      = packed array (.1..unitsize.) of char;
   operationstring = packed array (.1..operation_name_length.) of char;
   countrystring   = packed array (.1..country_name_length.) of char;
   missions        = (attack,defend);
   areas           = (conus,europe,korea);
   climates        = (hot,temperate,cold);
   intensities     = (hi,mid,low);
   tf_sizes        = (bn,bde);
   tf_types        = (armor,mech,inf);
   terrains        = (woods,open,built_up,mountains);
   durations       = (first_day,succeeding_day);
   update_sources  = (none,estimate,factual);
   visibilities    = (good,fair,poor);
   ration_policy_type = (b_c_b,c_c_b);  (*mix of ration types        *)
   analogy_data = record
     analogy_index : integer;   (*index into analogies used          *)
     date          : datestring;(*date of analogous operation        *)
     unit          : unitstring;(*unit name in analogous operation   *)
     quality_pts   : integer    (*measure of analogy strength        *)
   end;(*end record analogy_info*)
   analogy_array = array(.1..max_analogies.) of analogy_data;
   analogy_record = record
        num_analogies : integer;
        analogies     : analogy_array
   end;(*end record analogy_info*)
```

```
   oprecord = record
      date                : datestring;
      unit                : unitstring;
      mission             : missions;
      climate             : climates;
      area                : areas;
      tf_type             : tf_types;
      tf_size             : tf_sizes;
      intensity           : intensities;
      moppcondition       : boolean;
      personnel_strength  : integer;
      ration_policy       : ration_policy_type;
      AF_ground_spt       : boolean;
      country             : countrystring;
      terrain             : terrains;
      update_source       : update_sources;
      duration            : durations;
      visibility          : visibilities;
      operation_name      : operationstring;
      consumption         : consumearray;
      update              : boolean;
      analogy_info        : analogy_record
   end;(*end record oprecord*)
    historytype = array (.1..maxfiles.) of oprecord;
    hist_file   = file of oprecord;


var
   history      : historytype; (*array of operations in the history
                                  files                              *)
   history_file : hist_file;   (*secondary storage file of oprecords  *)
   file_counter : integer;     (*number of records in history file    *)
   module_code  : char;        (*user selection of program module     *)
   finished     : boolean;     (*flag to halt program execution       *)


procedure initialize;
var
   ok     : boolean;          (*validation of acceptable user choice  *)
   answer : char;             (*user response about historical records *)
begin
   page;
   finished:= false;
   writeln('This program is designed to assist the tactical unit');
   writeln('logistics planner at the battalion and brigade level.');
   writeln;writeln;writeln;
   write('Is there an already existing historical file of previous');
   writeln(' operations?');
   writeln;writeln;
   ok:= false;
   writeln('Enter the number corresponding to your answer.');
   writeln('    1 - yes, there is a file called hist hist_file a');
   writeln('    2 - no,  there is no historical file ');
   repeat
      readln(answer);
      writeln;
      writeln(answer);
      file_counter:= 0;
      if answer = '1' then
        begin
          ok:= true;
          reset(history_file,'hist oprecord a');
          while not eof(history_file) do
            begin
              file_counter:= file_counter + 1;
              read(history_file,history(.file_counter.))
            end;
        end
```

```
      else if answer = '2' then ok:= true
      else writeln('You have made an error in input, try again.')
    until ok = true;
    page
end; (*end procedure initialize*)


procedure module_choice;
 var
   ok              : boolean; (*validation of acceptable user choice    *)
   continue_char : char;    (*user response to continue with program  *)
begin
  writeln('There are ',file_counter:2,' records in the history files.');
  writeln('Under current program parameters, there is storage for  ');
  writeln(maxfiles-file_counter:2,' additional records.');
  writeln;
  writeln('Additional storage can be obtained by either deleting');
  writeln('already existing records from the history files or by');
  writeln('changing the program parameters.');
  writeln;writeln;
  writeln('Enter c to continue');
  repeat
    readln(continue_char)
  until continue_char = 'c';
  page;
  writeln('The program will perform the following tasks.');
  writeln('Enter the number corresponding to the desired function.');
  writeln;
  writeln(' 1 - create a logistics estimate for an operation.');
  writeln;
  writeln(' 2 - update the historical file of a previous operation');
  writeln('     with user supplied consumption data.');
  writeln;
  writeln(' 3 - delete the records pertaining to operations for');
  writeln('     which the user no longer has any use.') ;
  writeln;
  writeln(' 4 - print the historical files.');
  writeln;
  writeln(' 5 - print the directory.');
  writeln;
  writeln(' 6 - quit the program.');
  writeln;writeln;
  ok:= false;
  repeat
     readln(module_code);
     if (module_code = '1') or (module_code = '2') or
        (module_code = '3') or (module_code = '4') or
        (module_code = '5') or (module_code = '6') then
          begin
            ok:= true;
            writeln('The module selected was # ',module_code)
          end
     else begin
            writeln('you have made an error in input, try again.');
            writeln
          end;
   until ok = true;
   page;
end; (*end procedure module_choice*)
```

```
(*********************************************************************
                    MODULE LOGISTICS_ESTIMATE
 *********************************************************************)
procedure log_estimate;
const
   LIN_size           =  6;   (*the width of the line item number field*)
   enditem_name_size  = 19;   (*the width of the enditem name field    *)
   num_enditems       = 19;   (*the number of end items modelled in this
                                program                                 *)
type
    LINstring        = packed array (.1..LIN_size.) of char;
    enditem_nametype = packed array (.1..enditem_name_size.) of char;
    enditem = record
        LIN          : LINstring;
        nomenclature : enditem_nametype;
        quantity     : integer
    end;(*end record enditem*)
    compositiontype = array (.1..num_enditems.) of enditem;
var
   taskforce :compositiontype;  (*record containing all the information
                                  about the components of a task force*)
   newrecord : oprecord;        (*record containing all the information
                                  about an operation                    *)

procedure build_task_force;
var
   i : integer; (*index variable for task force items*)
begin
   writeln('You will now begin building the task force.');
   writeln;writeln;writeln;
   taskforce(.1.).LIN := 'J81750';
   taskforce(.1.).nomenclature:= 'M2 INF FIGHTING VEH';
   taskforce(.1.).quantity := 0;
   taskforce(.2.).LIN := 'C76335';
   taskforce(.2.).nomenclature:=  'M3 CAV FIGHTING VEH';
   taskforce(.2.).quantity := 0;
   taskforce(.3.).LIN := 'D12087';
   taskforce(.3.).nomenclature:=  'M113 PERS CARRIER  ';
   taskforce(.3.).quantity := 0;
   taskforce(.4.).LIN := 'E56896';
   taskforce(.4.).nomenclature:=  'M901 CBT VEH ITV   ';
   taskforce(.4.).quantity := 0;
   taskforce(.5.).LIN := 'D10726';
   taskforce(.5.).nomenclature:=  'M125A1 81MM CARR   ';
   taskforce(.5.).quantity := 0;
   taskforce(.6.).LIN := 'D10741';
   taskforce(.6.).nomenclature:=  'M106A1 107MM  CARR ';
   taskforce(.6.).quantity := 0;
   taskforce(.7.).LIN := 'XXXXXX';
   taskforce(.7.).nomenclature:=  'M102 105MM HOW     ';
   taskforce(.7.).quantity := 0;
   taskforce(.8.).LIN := 'K57667';
   taskforce(.8.).nomenclature:= 'M109 155MM SP HOW  ';
   taskforce(.8.).quantity := 0;
   taskforce(.9.).LIN := 'K56981';
   taskforce(.9.).nomenclature:= 'M110 8in SP HOW    ';
   taskforce(.9.).quantity := 0;
   taskforce(.10.).LIN := 'L44894';
   taskforce(.10.).nomenclature:= 'LAUN-LOAD MLRS     ';
   taskforce(.10.).quantity := 0;
   taskforce(.11.).LIN := 'J96694';
   taskforce(.11.).nomenclature:= 'M163 VULCAN AIR DEF';
   taskforce(.11.).quantity := 0;
   taskforce(.12.).LIN := 'D11668';
   taskforce(.12.).nomenclature:= 'M730 CHAP AIR DEF  ';
   taskforce(.12.).quantity := 0;
```

```
   taskforce(.13.).LIN := 'T13374';
   taskforce(.13.).nomenclature:= 'M1 TANK 105MM      ';
   taskforce(.13.).quantity := 0;
   taskforce(.14.).LIN := 'V13101';
   taskforce(.14.).nomenclature:= 'M60 TANK 105MM     ';
   taskforce(.14.).quantity := 0;
   taskforce(.15.).LIN := 'XXXXXX';
   taskforce(.15.).nomenclature:= 'TOW LAUNCHER       ';
   taskforce(.15.).quantity := 0;
   taskforce(.16.).LIN := 'XXXXXX';
   taskforce(.16.).nomenclature:= 'M222 DRAGON LNCHR  ';
   taskforce(.16.).quantity := 0;
   taskforce(.17.).LIN := 'XXXXXX';
   taskforce(.17.).nomenclature:= 'M2 50 CAL MG       ';
   taskforce(.17.).quantity := 0;
   taskforce(.18.).LIN := 'XXXXXX';
   taskforce(.18.).nomenclature:= 'M60 MG             ';
   taskforce(.18.).quantity := 0;
   taskforce(.19.).LIN := 'XXXXXX';
   taskforce(.19.).nomenclature:= 'M16A1 RIFLE        ';
   taskforce(.19.).quantity := 0;
    for i:= 1 to num_enditems do
      begin
        write('enter the number of ',taskforce(.i.).nomenclature);
        writeln(' in your task force.');
        readln(taskforce(.i.).quantity);
        writeln(taskforce(.i.).quantity)
      end;
    page;
    writeln('the task force has been built. task force composition is');
    writeln;writeln;writeln;
    for i:= 1 to num_enditems do
        begin
          write(taskforce(.i.).nomenclature,'     ');
          writeln(taskforce(.i.).quantity)
        end
end;(*end procedure buildtaskforce*)


procedure create_scenario;
procedure readdate;
var
  ok      : boolean;
  newdate : datestring;
  answer  : char;
begin
  ok:= false;
  repeat
    writeln('Enter the date on which the operation is to commence.');
    writeln('Use the form dd/mm/yy');
    readln(newdate);
    writeln;
    writeln('The date of the operation is ', newdate);
    writeln;
    writeln('Is this the correct date?');
    writeln('Enter the number corresponding to your answer.');
    writeln('    1 - yes, date is correct  ');
    writeln('    2 - no,  date is incorrect');
    readln(answer);
    if answer = '1' then
      begin
        newrecord.date:= newdate;
        ok:= true
      end
  until ok = true;
  writeln;writeln;writeln
end; (*end procedure readdate*)
```

```
procedure readunit;
const
   blanks = '       ';
var
   ok       : boolean;
   answer   : char;
   unitname : unitstring;
begin
   ok:= false;
   repeat
     writeln('Enter the name of the unit for which this estimate or ');
     writeln('update is being prepared. For example- 1/33rd ');
     readln(unitname);
     strconcat(unitname,blanks);
     writeln;
     writeln('The name of the unit is ', unitname);
     writeln;writeln;
     writeln('Is this the correct unit name?');
     writeln('Enter the number corresponding to your answer.');
     writeln('    1 - yes, unit name is correct  ');
     writeln('    2 - no,  unit name is incorrect');
     readln(answer);
     if answer = '1' then
       begin
         newrecord.unit:= unitname;
         ok:= true
       end
   until ok = true;
   writeln;writeln;writeln
end; (*end procedure readunit*)


procedure readmission;
var
   ok           : boolean;
   mission_code : char;
begin
   writeln('Enter the number corresponding to the correct mission.');
   writeln('    1 - attack  ');
   writeln('    2 - defend  ');
   ok:= false;
   repeat
      readln(mission_code);
      if mission_code = '1' then
         begin
           newrecord.mission := attack;
           ok:= true
         end
      else if mission_code = '2' then
         begin
           newrecord.mission:= defend;
           ok:= true
         end
      else writeln('you have made an error in input, try again.');
   until ok = true;
   writeln;writeln;writeln
end; (*end procedure readmission*)


procedure readclimate;
var
   ok           : boolean;
   climate_code : char;
begin
   writeln('Enter the number corresponding to the correct climate.');
   writeln('    1 - hot         ');
   writeln('    2 - temperate   ');
   writeln('    3 - cold        ');
```

```
   ok:= false;
   repeat
      readln(climate_code);
      if climate_code = '1' then
         begin
           newrecord.climate := hot;
           ok:= true
         end
      else if climate_code = '2' then
         begin
           newrecord.climate:= temperate;
           ok:= true
         end
      else if climate_code = '3' then
         begin
           newrecord.climate := cold;
           ok:= true
         end
      else writeln('you have made an error in input, try again.');
   until ok = true;
   writeln;writeln;writeln
end; (*end procedure readclimate*)


procedure readarea;
var
   ok        : boolean;
   area_code : char;
begin
   writeln('Enter the number corresponding to the correct area.');
   writeln('    1 - conus   ');
   writeln('    2 - europe  ');
   writeln('    3 - korea   ');
   ok:= false;
   repeat
      readln(area_code);
      if area_code = '1' then
         begin
           newrecord.area := conus;
           ok:= true
         end
      else if area_code = '2' then
         begin
           newrecord.area:= europe;
           ok:= true
         end
      else if area_code = '3' then
         begin
           newrecord.area:= korea;
           ok:= true
         end
      else writeln('you have made an error in input, try again.');
   until ok = true;
   writeln;writeln;writeln
end; (*end procedure readarea*)


procedure readtftype;
var
   ok          : boolean;
   tftype_code : char;
begin
   writeln('Enter the number corresponding to the correct tf type.');
   writeln('    1 - armor      ');
   writeln('    2 - mechanized ');
   writeln('    3 - infantry   ');
   ok:= false;
   repeat
      readln(tftype_code);
```

```
      if tftype_code = '1' then
         begin
           newrecord.tf_type := armor;
           ok:= true
         end
      else if tftype_code = '2' then
         begin
           newrecord.tf_type:= mech;
           ok:= true
         end
      else if tftype_code = '3' then
         begin
           newrecord.tf_type := inf;
           ok:= true
         end
      else writeln('you have made an error in input, try again.');
   until ok = true;
   writeln;writeln;writeln
end; (*end procedure readtftype*)


procedure readtfsize;
var
   ok          : boolean;
   tfsize_code : char;
begin
   writeln('Enter the number corresponding to the correct tf size.');
   writeln('    1 - battalion  ');
   writeln('    2 - brigade    ');
   ok:= false;
   repeat
      readln(tfsize_code);
      if tfsize_code = '1' then
         begin
           newrecord.tf_size := bn;
           ok:= true
         end
      else if tfsize_code = '2' then
         begin
           newrecord.tf_size:= bde;
           ok:= true
         end
      else writeln('you have made an error in input, try again.');
   until ok = true;
   writeln;writeln;writeln
end; (*end procedure readtfsize*)


procedure readintensity;
var
   ok             : boolean;
   intensity_code : char;
begin
   writeln('Enter the number corresponding to the correct intensity');
   writeln('    1 - high  ');
   writeln('    2 - mid   ');
   writeln('    3 - low   ');
   ok:= false;
   repeat
      readln(intensity_code);
      if intensity_code = '1' then
         begin
           newrecord.intensity:= hi;
           ok:= true
         end
      else if intensity_code = '2' then
         begin
           newrecord.intensity:= mid;
           ok:= true
```

```
            end
         else if intensity_code = '3' then
            begin
               newrecord.intensity:= low;
               ok:= true
            end
         else writeln('you have made an error in input, try again.');
   until ok = true;
   writeln;writeln;writeln
end; (*end procedure readintensity*)


procedure readmopp;
var
   ok     : boolean;
   answer : char;
begin
   writeln('Do you expect the task force to be in MOPP level three');
   writeln('or MOPP level four during this mission.');
   writeln;
   writeln('Enter the correct number for your response');
   writeln('   1 - yes');
   writeln('   2 - no ');
   ok:= false;
   repeat
      readln(answer);
      if answer = '1' then
         begin
            newrecord.moppcondition:= true;
            ok:= true
         end
      else if answer = '2' then
         begin
            newrecord.moppcondition:= false;
            ok:= true
         end
      else writeln('You have made an error in input, try again.');
   until ok = true;
   writeln;writeln;writeln
end; (*end procedure readmopp*)


procedure readpersonnel;
const
   maxpersonnel = 10000;
var
   ok           : boolean;
   numpersonnel : integer;
begin
   ok:= false;
   repeat
      writeln('Enter the total number of personnel in the task force.');
      readln(numpersonnel);
      if (numpersonnel > 0) and (numpersonnel < maxpersonnel) then
         begin
            newrecord.personnel_strength:= numpersonnel;
            ok:= true
         end
      else begin
                write('The number of personnel exceeds program parameters.');
                writeln('  Input the number again.')
             end
   until ok = true;
   writeln;writeln;writeln
end; (*end procedure readpersonnel*)
```

```
procedure readterrain;
var
   ok           : boolean;
   terrain_code : char;
begin
   writeln('Enter the number corresponding to the correct terrain');
   writeln('    1 - open  ');
   writeln('    2 - woods ');
   writeln('    3 - built up ');
   writeln('    4 - mountainous ');
   ok:= false;
   repeat
       readln(terrain_code);
       if terrain_code = '1' then
          begin
            newrecord.terrain:= open;
            ok:= true
          end
       else if terrain_code = '2' then
          begin
            newrecord.terrain:= woods;
            ok:= true
          end
       else if terrain_code = '3' then
          begin
            newrecord.terrain:= built_up;
            ok:= true
          end
       else if terrain_code = '4' then
          begin
            newrecord.terrain:= mountains;
            ok:= true
          end
       else writeln('you have made an error in input, try again.');
   until ok = true;
   writeln;writeln;writeln
end; (*end procedure readterrain*)


procedure readvisibility;
var
   ok              : boolean;
   visibility_code : char;
begin
   writeln('Enter the number corresponding to the visibility');
   writeln('    1 - good  ');
   writeln('    2 - fair  ');
   writeln('    3 - poor  ');
   ok:= false;
   repeat
       readln(visibility_code);
       if visibility_code = '1' then
          begin
            newrecord.visibility:= good;
            ok:= true
          end
       else if visibility_code = '2' then
          begin
            newrecord.visibility:= fair;
            ok:= true
          end
       else if visibility_code = '3' then
          begin
            newrecord.visibility:= poor;
            ok:= true
          end
       else writeln('you have made an error in input, try again.');
   until ok = true;
   writeln;writeln;writeln
```

```
end; (*end procedure readvisibility*)


procedure readAF_ground_spt;
var
   ok     : boolean;
   answer : char;
begin
   writeln('Do you plan on significant Air Force ground support?');
   writeln;
   writeln('Enter the correct number for your response');
   writeln('   1 - yes');
   writeln('   2 - no ');
   ok:= false;
   repeat
      readln(answer);
      if answer = '1' then
        begin
          newrecord.AF_ground_spt:= true;
          ok:= true
        end
      else if answer = '2' then
        begin
          newrecord.AF_ground_spt:= false;
          ok:= true
        end
      else writeln('You have made an error in input, try again.');
   until ok = true;
   writeln;writeln;writeln
end; (*end procedure readAF_ground_spt*)


procedure readduration;
var
   ok     : boolean;
   answer : char;
begin
   writeln('Is this the first day of this mission or is this a');
   writeln('succeeding day of a continuing mission.');
   writeln;
   writeln('Enter the correct number for your response');
   writeln('   1 - first day');
   writeln('   2 - succeeding day ');
   ok:= false;
   repeat
      readln(answer);
      if answer = '1' then
        begin
          newrecord.duration:= first_day;
          ok:= true
        end
      else if answer = '2' then
        begin
          newrecord.duration:= succeeding_day;
          ok:= true
        end
      else writeln('You have made an error in input, try again.');
   until ok = true;
   writeln;writeln;writeln
end; (*end procedure readduration*)


procedure readoperation_name;
const
   blanks = '       ';
var
   ok              : boolean;
   answer          : char;
```

```
  operation_name : operationstring;
begin
  ok:= false;
  repeat
    writeln('Enter the name of the operation of which this mission ');
    writeln('is a part.  For example- D-DAY ');
    readln(operation_name);
    strconcat(operation_name,blanks);
    writeln;
    writeln('The name of the operation is ', operation_name);
    writeln;writeln;
    writeln('Is this correct ?');
    writeln('Enter the number corresponding to your answer.');
    writeln('    1 - yes, operation name is correct  ');
    writeln('    2 - no,  operation name is incorrect');
    readln(answer);
    if answer = '1' then
      begin
        newrecord.operation_name:= operation_name;
        ok:= true
      end
  until ok = true;
  writeln;writeln;writeln
end; (*end procedure readoperation_name*)


procedure readcountry_name;
const
  blanks = '       ';
var
  ok           : boolean;
  answer       : char;
  country_name : countrystring;
begin
  ok:= false;
  repeat
    writeln('Enter the name of the country in which this mission ');
    writeln('will be conducted.  For example- West Germany.');
    writeln('Be sure to capitalize the first letter in each word');
    readln(country_name);
    strconcat(country_name,blanks);
    writeln;
    writeln('The name of the country is ', country_name);
    writeln;writeln;
    writeln('Is this correct ?');
    writeln('Enter the number corresponding to your answer.');
    writeln('    1 - yes, country name is correct  ');
    writeln('    2 - no,  country name is incorrect');
    readln(answer);
    if answer = '1' then
      begin
        newrecord.country:= country_name;
        ok:= true
      end
  until ok = true;
  writeln;writeln;writeln
end; (*end procedure readcountry_name*)


procedure readrationpolicy;
var
  ok     : boolean;
  answer : char;
begin
  writeln('Enter the number corresponding to the ration policy');
  writeln('during the duration of this operation.');
  writeln;writeln;
  writeln('   1 - b_c_b');
  writeln('   2 - c_c_b');
```

```
   ok:= false;
   repeat
     readln(answer);
     if answer = '1' then
       begin
         newrecord.ration_policy:= b_c_b;
         ok:= true
       end
     else if answer = '2' then
       begin
         newrecord.ration_policy:= c_c_b;
         ok:= true
       end
     else writeln('You have made an error in input, try again.');
   until ok = true;
   writeln
end; (*end procedure readrationpolicy*)


procedure buildconsarray;
var
   i : integer; (*loop control variable*)
begin
   newrecord.consumption(.1.).supply_item := 'water               ';
   newrecord.consumption(.1.).unit_of_measure := 'gallons';
   newrecord.consumption(.2.).supply_item := 'B rations           ';
   newrecord.consumption(.2.).unit_of_measure := 'meals  ';
   newrecord.consumption(.3.).supply_item := 'MRE rations         ';
   newrecord.consumption(.3.).unit_of_measure := 'meals  ';
   newrecord.consumption(.4.).supply_item := 'class II supplies  ';
   newrecord.consumption(.4.).unit_of_measure := 'STONS  ';
   newrecord.consumption(.5.).supply_item := 'diesel fuel         ';
   newrecord.consumption(.5.).unit_of_measure := 'gallons';
   newrecord.consumption(.6.).supply_item := 'class IV supplies  ';
   newrecord.consumption(.6.).unit_of_measure := 'STONS  ';
   newrecord.consumption(.7.).supply_item := 'tank ammo 105mm    ';
   newrecord.consumption(.7.).unit_of_measure := 'rounds ';
   newrecord.consumption(.8.).supply_item := 'TOW ammo            ';
   newrecord.consumption(.8.).unit_of_measure := 'rounds ';
   newrecord.consumption(.9.).supply_item := 'DRAGON ammo         ';
   newrecord.consumption(.9.).unit_of_measure := 'rounds ';
   newrecord.consumption(.10.).supply_item := 'Howitzer ammo 105mm';
   newrecord.consumption(.10.).unit_of_measure := 'rounds ';
   newrecord.consumption(.11.).supply_item := 'Howitzer ammo 155mm';
   newrecord.consumption(.11.).unit_of_measure := 'rounds ';
   newrecord.consumption(.12.).supply_item := 'Howitzer ammo 8in  ';
   newrecord.consumption(.12.).unit_of_measure := 'rounds ';
   newrecord.consumption(.13.).supply_item := 'Vulcan ammo 20mm   ';
   newrecord.consumption(.13.).unit_of_measure := 'rounds ';
   newrecord.consumption(.14.).supply_item := 'Mortar ammo 81mm   ';
   newrecord.consumption(.14.).unit_of_measure := 'rounds ';
   newrecord.consumption(.15.).supply_item := 'Mortar ammo 107mm  ';
   newrecord.consumption(.15.).unit_of_measure := 'rounds ';
   newrecord.consumption(.16.).supply_item := 'MG ammo .50 caliber';
   newrecord.consumption(.16.).unit_of_measure := 'rounds ';
   newrecord.consumption(.17.).supply_item := 'MG ammo 7.62mm     ';
   newrecord.consumption(.17.).unit_of_measure := 'rounds ';
   newrecord.consumption(.18.).supply_item := 'rifle ammo 5.56mm  ';
   newrecord.consumption(.18.).unit_of_measure := 'rounds ';
   newrecord.consumption(.19.).supply_item := 'class VII supplies ';
   newrecord.consumption(.19.).unit_of_measure := 'STONS  ';
   newrecord.consumption(.20.).supply_item := 'class VIII supplies';
   newrecord.consumption(.20.).unit_of_measure := 'STONS  ';
   newrecord.consumption(.21.).supply_item := 'class IX supplies  ';
   newrecord.consumption(.21.).unit_of_measure := 'STONS  ';
   for i:= 1 to num_supply_items do
     newrecord.consumption(.i.).actual_consumption := 0
end; (*end procedure buildconsarray*)
```

```
begin (*begin of create scenario*)
   page;
   writeln('The following questions describe the operation for which ');
   writeln('the program will create a logistics requirements estimate.');
   writeln('All questions must be answered as directed.');
   writeln;writeln;writeln;
   readdate;
   readunit;
   readtftype;
   readtfsize;
   readmission;
   readduration;
   readoperation_name;
   readarea;
   readcountry_name;
   readclimate;
   readintensity;
   readmopp;
   readterrain;
   readvisibility;
   readAF_ground_spt;
   readpersonnel;
   readrationpolicy;
   buildconsarray;
   newrecord.update_source:= none;
   newrecord.update := false;
end;(*end procedure create_scenario*)


procedure create_estimate;
var
   i : integer;        (*index variable for input into consarray*)
procedure water_estimate;
var
   drinking_requirements : real;
   heat_treatment        : real;
   personal_hygiene      : real;
   food_preparation      : real;
begin
   case newrecord.climate of
            hot  : begin
                        if newrecord.moppcondition = true then
                              drinking_requirements:= 3.5
                        else drinking_requirements:= 3.0;
                        heat_treatment:= 0.2;
                        personal_hygiene:= 0.7;
                        if newrecord.ration_policy = b_c_b then
                              food_preparation:= 1.0
                        else food_preparation:= 0.5
                      end;
       temperate : begin
                        if newrecord.moppcondition = true then
                           drinking_requirements:= 3.0
                        else drinking_requirements:= 1.5;
                        heat_treatment:= 0.0;
                        personal_hygiene:= 0.7;
                        if newrecord.ration_policy = b_c_b then
                           food_preparation:= 1.0
                        else food_preparation:= 0.5
                      end;
            cold  : begin
                        if newrecord.moppcondition = true then
                              drinking_requirements:= 2.0
                        else drinking_requirements:= 2.0;
                        heat_treatment:= 0.0;
                        personal_hygiene:= 0.7;
                        if newrecord.ration_policy = b_c_b then
```

```
                        food_preparation:= 1.0
                      else food_preparation:= 0.5
                    end
   end; (*end case statement*)
   newrecord.consumption(.1.).general_estimate:=
   round((drinking_requirements + heat_treatment + personal_hygiene +
        food_preparation)* 1.10 * newrecord.personnel_strength)
end; (*end procedure water_estimate*)


procedure class_I_estimate;
begin
   if newrecord.ration_policy = b_c_b then
     begin
       newrecord.consumption(.2.).general_estimate:=
         (newrecord.personnel_strength * 2);
       newrecord.consumption(.3.).general_estimate:=
         newrecord.personnel_strength
     end
   else
     begin
       newrecord.consumption(.3.).general_estimate:=
         newrecord.personnel_strength * 2;
       newrecord.consumption(.2.).general_estimate:=
         newrecord.personnel_strength
     end
end; (*end procedure class_I_estimate*)


procedure compute_general_supplies(consumption_array_index:integer;
                                   consumption_factor     :real);
begin
   newrecord.consumption(.consumption_array_index.).general_estimate:=
   round((newrecord.personnel_strength * consumption_factor) / 2000)
end; (*end procedure compute_general_supplies*)


procedure diesel_fuel_estimate;
(*general formula used = for each weapon , take the sum of the following
                    # weapons * #hrs_idle   * consumption_idle +
                    # weapons * #hrs_xcntry * consumption_xcntry +
                    # weapons * #hrs_2ndrds * consumption_2ndrds.
   Then sum all of these for total diesel fuel required.
   Note: the fuel estimate for 105mm towed howitzer is for a M35 vehicle
         operating 24 hours                                              *)
begin
   case newrecord.area of
     korea :  newrecord.consumption(.5.).general_estimate:=
                round(taskforce(.1.).quantity * 3.0 *  6.4 +
                      taskforce(.1.).quantity * 5.5 * 18.0 +
                      taskforce(.1.).quantity * 5.5 *  8.6 +
                      taskforce(.2.).quantity * 3.0 *  6.4 +
                      taskforce(.2.).quantity * 5.5 * 18.0 +
                      taskforce(.2.).quantity * 5.5 *  8.6 +
                      taskforce(.3.).quantity * 3.1 *  1.0 +
                      taskforce(.3.).quantity  * 5.5 *  8.6 +
                      taskforce(.3.).quantity  * 5.5 * 10.3 +
                      taskforce(.4.).quantity  * 3.0 *  1.0 +
                      taskforce(.4.).quantity  * 5.5 *  8.6 +
                      taskforce(.4.).quantity  * 5.5 *  8.9 +
                      taskforce(.5.).quantity  * 4.1 *  1.0 +
                      taskforce(.5.).quantity  * 5.0 *  8.6 +
                      taskforce(.5.).quantity  * 5.0 * 10.3 +
                      taskforce(.6.).quantity  * 4.1 *  1.0 +
                      taskforce(.6.).quantity  * 5.0 * 10.0 +
                      taskforce(.6.).quantity  * 5.0 * 13.3 +
                      taskforce(.7.).quantity  *24.0 *  0.2 +
                      taskforce(.8.).quantity  * 4.1 *  1.0 +
```

```
                    taskforce(.8.).quantity  * 6.0 * 11.8 +
                    taskforce(.8.).quantity  * 5.5 * 16.1 +
                    taskforce(.9.).quantity  * 6.5 *  1.6 +
                    taskforce(.9.).quantity  * 3.6 * 12.5 +
                    taskforce(.9.).quantity  * 6.0 * 14.3 +
                    taskforce(.10.).quantity * 5.0 *  1.0 +
                    taskforce(.10.).quantity * 4.0 *  6.2 +
                    taskforce(.10.).quantity * 4.5 *  8.9 +
                    taskforce(.11.).quantity * 4.1 *  1.0 +
                    taskforce(.11.).quantity * 6.0 *  5.2 +
                    taskforce(.11.).quantity * 5.5 * 13.0 +
                    taskforce(.12.).quantity * 4.0 *  0.5 +
                    taskforce(.12.).quantity * 6.0 *  1.3 +
                    taskforce(.12.).quantity * 5.5 *  2.6 +
                    taskforce(.13.).quantity * 5.2 * 10.8 +
                    taskforce(.13.).quantity * 3.3 * 56.6 +
                    taskforce(.13.).quantity * 3.4 * 44.7 +
                    taskforce(.14.).quantity * 4.6 *  2.0 +
                    taskforce(.14.).quantity * 6.5 * 28.1 +
                    taskforce(.14.).quantity * 4.6 * 35.7 );
europe : newrecord.consumption(.5.).general_estimate:=
           round(taskforce(.1.).quantity  * 3.0 *  6.4 +
                    taskforce(.1.).quantity  * 5.5 * 18.0 +
                    taskforce(.1.).quantity  * 5.5 *  8.6 +
                    taskforce(.2.).quantity  * 3.0 *  6.4 +
                    taskforce(.2.).quantity  * 5.5 * 18.0 +
                    taskforce(.2.).quantity  * 5.5 *  8.6 +
                    taskforce(.3.).quantity  * 3.0 *  1.0 +
                    taskforce(.3.).quantity  * 5.5 *  8.6 +
                    taskforce(.3.).quantity  * 5.5 * 10.3 +
                    taskforce(.4.).quantity  * 3.0 *  1.0 +
                    taskforce(.4.).quantity  * 5.5 *  8.6 +
                    taskforce(.4.).quantity  * 5.5 *  8.9 +
                    taskforce(.5.).quantity  * 4.0 *  1.0 +
                    taskforce(.5.).quantity  * 5.0 *  8.6 +
                    taskforce(.5.).quantity  * 5.0 * 10.3 +
                    taskforce(.6.).quantity  * 4.0 *  1.0 +
                    taskforce(.6.).quantity  * 5.0 * 10.0 +
                    taskforce(.6.).quantity  * 5.0 * 13.3 +
                    taskforce(.7.).quantity  *24.0 *  0.2 +
                    taskforce(.8.).quantity  * 4.0 *  1.0 +
                    taskforce(.8.).quantity  * 6.0 * 11.8 +
                    taskforce(.8.).quantity  * 5.5 * 16.1 +
                    taskforce(.9.).quantity  * 4.0 *  1.6 +
                    taskforce(.9.).quantity  * 6.0 * 12.5 +
                    taskforce(.9.).quantity  * 5.5 * 14.3 +
                    taskforce(.10.).quantity * 5.0 *  1.0 +
                    taskforce(.10.).quantity * 5.0 *  6.2 +
                    taskforce(.10.).quantity * 4.5 *  8.9 +
                    taskforce(.11.).quantity * 4.0 *  1.0 +
                    taskforce(.11.).quantity * 6.0 *  5.2 +
                    taskforce(.11.).quantity * 5.5 * 13.0 +
                    taskforce(.12.).quantity * 4.0 *  0.5 +
                    taskforce(.12.).quantity * 6.0 *  1.3 +
                    taskforce(.12.).quantity * 5.5 *  2.6 +
                    taskforce(.13.).quantity * 5.0 * 10.8 +
                    taskforce(.13.).quantity * 6.5 * 56.6 +
                    taskforce(.13.).quantity * 5.0 * 44.7 +
                    taskforce(.14.).quantity * 4.5 *  2.0 +
                    taskforce(.14.).quantity * 6.5 * 28.1 +
                    taskforce(.14.).quantity * 4.5 * 35.7 );
conus :  newrecord.consumption(.5.).general_estimate:=
           round(taskforce(.1.).quantity  * 3.0 *  6.4 +
                    taskforce(.1.).quantity  * 5.5 * 18.0 +
                    taskforce(.1.).quantity  * 5.5 *  8.6 +
                    taskforce(.2.).quantity  * 3.0 *  6.4 +
                    taskforce(.2.).quantity  * 5.5 * 18.0 +
                    taskforce(.2.).quantity  * 5.5 *  8.6 +
                    taskforce(.3.).quantity  * 7.0 *  1.0 +
                    taskforce(.3.).quantity  * 6.8 *  8.6 +
```

```
                        taskforce(.3.).quantity  * 1.9 * 10.3 +
                        taskforce(.4.).quantity  * 3.0 *  1.0 +
                        taskforce(.4.).quantity  * 5.5 *  8.6 +
                        taskforce(.4.).quantity  * 5.5 *  8.9 +
                        taskforce(.5.).quantity  * 5.0 *  1.0 +
                        taskforce(.5.).quantity  * 3.8 *  8.6 +
                        taskforce(.5.).quantity  * 1.6 * 10.3 +
                        taskforce(.6.).quantity  * 5.3 *  1.0 +
                        taskforce(.6.).quantity  * 3.1 * 10.0 +
                        taskforce(.6.).quantity  * 4.3 * 13.3 +
                        taskforce(.7.).quantity  *24.0 *  0.2 +
                        taskforce(.8.).quantity  * 6.2 *  1.0 +
                        taskforce(.8.).quantity  * 1.9 * 11.8 +
                        taskforce(.8.).quantity  * 2.9 * 16.1 +
                        taskforce(.9.).quantity  * 4.1 *  1.6 +
                        taskforce(.9.).quantity  * 1.9 * 12.5 +
                        taskforce(.9.).quantity  * 4.1 * 14.3 +
                        taskforce(.10.).quantity * 5.0 *  1.0 +
                        taskforce(.10.).quantity * 4.0 *  6.2 +
                        taskforce(.10.).quantity * 4.5 *  8.9 +
                        taskforce(.11.).quantity * 2.4 *  1.0 +
                        taskforce(.11.).quantity * 7.2 *  5.2 +
                        taskforce(.11.).quantity * 4.8 * 13.0 +
                        taskforce(.12.).quantity * 4.0 *  0.5 +
                        taskforce(.12.).quantity * 6.0 *  1.3 +
                        taskforce(.12.).quantity * 5.5 *  2.6 +
                        taskforce(.13.).quantity * 5.2 * 10.8 +
                        taskforce(.13.).quantity * 3.3 * 56.6 +
                        taskforce(.13.).quantity * 3.4 * 44.7 +
                        taskforce(.14.).quantity * 4.2 *  2.0 +
                        taskforce(.14.).quantity * 8.5 * 28.1 +
                        taskforce(.14.).quantity * 2.9 * 35.7 )
  end; (*end case statement*)
end; (*end procedure diesel_fuel_estimate*)


procedure compute_ammo(cons_num,num_weapons,ha,hd,ma,md,la,ld:integer);
begin
  case newrecord.intensity of
     hi :case newrecord.mission of
           attack : newrecord.consumption(.cons_num.).general_estimate:=
                      num_weapons * ha;
           defend : newrecord.consumption(.cons_num.).general_estimate:=
                      num_weapons * hd;
         end;
    mid :case newrecord.mission of
           attack : newrecord.consumption(.cons_num.).general_estimate:=
                      num_weapons * ma;
           defend : newrecord.consumption(.cons_num.).general_estimate:=
                      num_weapons * md;
         end;
    low :case newrecord.mission of
           attack : newrecord.consumption(.cons_num.).general_estimate:=
                      num_weapons * la;
           defend : newrecord.consumption(.cons_num.).general_estimate:=
                      num_weapons * ld;
         end
  end (*end case statement*)
end;  (*end procedure compute_ammo*)


procedure adjust_estimate;
const
  max_candidates = 10;        (*max number of candidate analogies    *)
type
  candidate_info = record
     index_num : integer;     (*index into history array of records   *)
     strength_pts  : integer; (*measure of analogy strength           *)
```

```
      used          : boolean  (*used in adjustment analogy selection  *)
   end;(*end record candidate_info*)
   candidates = array (.1..max_candidates.) of candidate_info;
var
   i,                                (*loop control var into analogies    *)
   adjustment_index,                 (*loop control var into consumption  *)
   num_candidates,                   (*number of analogous records        *)
   index              :integer;      (*index into the history files       *)
   analogy_candidate : candidates;(*candidate records for adjustment   *)


function analogous(index:integer) : boolean;
begin
   with history(.index.) do
     begin
       if (update          = true) and
          (moppcondition  = newrecord.moppcondition) and
          (mission        = newrecord.mission) and
          (intensity      = newrecord.intensity) and
          (climate        = newrecord.climate) and
          (area           = newrecord.area) and
          (duration       = newrecord.duration) then analogous:= true
       else analogous:= false;
     end
end; (*end function analogous*)


procedure adjust (i: integer);
var
   sum_error       : real;          (*sum of errors in analogies used *)
   sum_quality_pts : real;          (*sum qlty  pts in analogies used *)
   analogy_count   : integer;       (*loop control variable           *)
begin
   sum_error:= 0;
   sum_quality_pts:=0;
   for analogy_count := 1 to newrecord.analogy_info.num_analogies do
     if history(.newrecord.analogy_info.analogies(.analogy_count.).
       analogy_index.).consumption(.i.).general_estimate > 0 then
     begin
       sum_quality_pts:= sum_quality_pts +
            newrecord.analogy_info.analogies(.analogy_count.).quality_pts;
       sum_error:= sum_error +
          (newrecord.analogy_info.analogies(.analogy_count.).quality_pts
         *((history(.newrecord.analogy_info.analogies(.analogy_count.).
              analogy_index.).consumption(.i.).actual_consumption -
            history(.newrecord.analogy_info.analogies(.analogy_count.).
              analogy_index.).consumption(.i.).general_estimate) /
            history(.newrecord.analogy_info.analogies(.analogy_count.).
              analogy_index.).consumption(.i.).general_estimate))
     end;
   newrecord.consumption(.i.).adjustments:=
     round ( newrecord.consumption(.i.).general_estimate *
           (sum_error / sum_quality_pts))
end; (*end procedure adjust*)


function compute_strength(af,vis,ter,update,cntry,unit,opname,af_wt,
                          vis_wt,ter_wt,update_wt,cntry_wt,unit_wt,
                          opname_wt,index : integer):integer;
var
   total_pts : integer;          (*total number of strength points      *)
begin
   total_pts:= 0;
   if (unit = 1) and (newrecord.unit = history(.index.).unit)
       then total_pts:= total_pts + unit_wt;
   if (update = 1) and (history(.index.).update_source = factual)
       then total_pts:= total_pts + update_wt;
```

```
   if (cntry = 1) and (newrecord.country = history(.index.).country)
      then total_pts:= total_pts + cntry_wt;
   if (ter = 1) and (newrecord.terrain = history(.index.).terrain)
      then total_pts:= total_pts + ter_wt;
   if (opname = 1) and (newrecord.operation_name = history(.index.)
      .operation_name) then total_pts:= total_pts + opname_wt;
   if (af  = 1) and (newrecord.AF_ground_spt = history(.index.)
      .AF_ground_spt) then total_pts:= total_pts + af_wt;
   if (vis = 1) and (newrecord.visibility = history(.index.).visibility)
      then total_pts:= total_pts + vis_wt;
   compute_strength:= total_pts
end;(*end function compute_strength*)


function pick_best_analogy:integer;
var
   strongest_analogy,           (*index of strongest analogy         *)
   analogy_candidate_num,       (*index into analogy_candidate array *)
   j,                           (*loop control variable              *)
   max_strength      : integer; (*maximium analogy strength          *)
begin
   strongest_analogy:= 0;
   max_strength:= -1;
   for j:= 1 to num_candidates do
     begin
       if (analogy_candidate(.j.).used = false) and
            (analogy_candidate(.j.).strength_pts > max_strength) then
          begin
            max_strength         := analogy_candidate(.j.).strength_pts;
            strongest_analogy    := analogy_candidate(.j.).index_num;
            analogy_candidate_num:= j
          end;
     end;
   analogy_candidate(.analogy_candidate_num.).used:= true;
   pick_best_analogy:= strongest_analogy;
end;(*end function pick_best_analogy*)


procedure print_analogies;
var
   i : integer;                 (*loop control variable              *)
begin
   page;
   writeln;writeln;writeln;writeln;
   writeln('                    ANALOGY REASONING');
   writeln;writeln;writeln;writeln;writeln;
   writeln('All of the available data on past operations has been ');
   writeln('evaluated to identify analogies to the current operation.');
   writeln;
   writeln('A previous operation is considered analogous to the');
   write('current operation if the following conditions are ');
   writeln('satisfied:');
   writeln;
   writeln('   1. The historical record of the previous operation has');
   writeln('      been updated with actual consumption data.');
   writeln;
   writeln('   2. Both operations have the same mission.');
   writeln;
   write('   3. Both operations took place in the same area ');
   writeln('of the world.');
   writeln;
   writeln('   4. Both operations took place in the same climate.');
   writeln;
   writeln('   5. Both operations took place under the same chemical');
   writeln('      defense mission oriented protective posture.');
   writeln;
   writeln('   6. Both operations involved the same combat intensity.');
   writeln;
```

```
   writeln('   7. Both operations were first day engagements or');
   writeln('      succeeding day engagements of the same mission type.');
   writeln;writeln;writeln;writeln;writeln;
   write('The following operations are analogous under this ');
   writeln('definition.');
   writeln;writeln;writeln;
   write('-------------------------------------------------------');
   writeln('-----------------------------------');
   write('  DATE      UNIT     MISSION    AREA     CLIMATE    MOPP  ');
   writeln('INTENSITY   FIRST/SUCCEEDING DAY');
   write('-------------------------------------------------------');
   writeln('-----------------------------------');
   writeln;
   if num_candidates > 0 then
     begin
         for i:= 1 to file_counter do
           if analogous(i) then
             begin
               write(history(.i.).date,'   ');
               write(history(.i.).unit);
               case history(.i.).mission of
                 attack : write('ATTACK    ');
                 defend : write('DEFEND    ');
               end;
               case history(.i.).area of
                 conus  : write('CONUS     ');
                 europe : write('EUROPE    ');
                 korea  : write('KOREA     ');
               end; (*end case statement*)
               case history(.i.).climate of
                 hot         :  write('HOT          ');
                 temperate   :  write('TEMPERATE    ');
                 cold        :  write('COLD         ')
               end;(*end case statement*)
               if history(.i.).moppcondition = true then
                  write('YES      ')
               else write('NO       ');
               case history(.i.).intensity of
                 hi      : write('HIGH         ');
                 mid     : write('MID          ');
                 low     : write('LOW          ');
               end; (*end case statement*)
               case history(.i.).duration of
                 first_day      :  writeln('FIRST DAY');
                 succeeding_day :  writeln('SUCCEEDING DAY');
               end;(*end case statement*)
               writeln
             end
     end
   else begin
           writeln;writeln;writeln;
           write('There are no analogous operations in the history ');
           writeln('in the history files.')
         end;
end; (*end procedure print_analogies*)


procedure print_water_reasoning;
var
   i : integer;                    (*loop control variable              *)
begin
   page;
   writeln;writeln;writeln;writeln;
   writeln('                    WATER SUPPLY REASONING');
   writeln;writeln;writeln;writeln;writeln;
   writeln('The analogous operations are evaluated on the strength');
   writeln('of their similarity to the current operation in those');
   writeln('areas pertinent to water supply consumption. Each of the');
   writeln('points of similarity are weighted independently.');
```

```
   writeln;
   writeln('The weighting of each item is in parenthesis below the ');
   writeln('item name. i.e. (3) = 3 points for AF_GROUND_SUPPORT');
   writeln;
   writeln('Up to three previous operations are considered in the ');
   writeln('adjustment algorithm, with those operations with the ');
   writeln('highest number of quality points being chosen for this ');
   writeln('purpose.');
   writeln;writeln;writeln;writeln;writeln;
   writeln('-------------------------------------');
   writeln('  DATE    UNIT       COUNTRY   UPDATE ');
   writeln('                       NAME     SOURCE ');
   writeln('-------------------------------------');
   writeln;
   writeln('                        (2)       (1)   ');
   writeln('-------------------------------------');
   writeln;writeln;
   for i:= 1 to num_candidates do
     begin
       write(history(.analogy_candidate(.i.).index_num.).date,'   ');
       write(history(.analogy_candidate(.i.).index_num.).unit,'  ');
       if history(.analogy_candidate(.i.).index_num.).country =
          newrecord.country then write('YES       ')
       else write(' NO       ');
       if history(.analogy_candidate(.i.).index_num.).update_source =
          factual then writeln('YES')
       else writeln(' NO');
       writeln
     end;(*end for loop*)
end;(*end procedure print_water_reasoning*)


procedure print_subsistence_reasoning;
var
   i : integer;                    (*loop control variable              *)
begin
   page;
   writeln;writeln;writeln;writeln;writeln;
   writeln('                  SUBSISTENCE SUPPLY REASONING');
   writeln;writeln;writeln;writeln;
   writeln('The analogous operations are evaluated on the strength');
   writeln('of their similarity to the current operation in those');
   writeln('areas pertinent to subsistence  consumption. Each of the');
   writeln('points of similarity are weighted independently.');
   writeln;
   writeln('The weighting of each item is in parenthesis below the ');
   writeln('item name. i.e. (3) = 3 points for AF_GROUND_SUPPORT');
   writeln;
   writeln('Up to three previous operations are considered in the ');
   writeln('adjustment algorithm, with those operations with the ');
   writeln('highest number of quality points being chosen for this ');
   writeln('purpose.');
   writeln;writeln;writeln;writeln;writeln;
   writeln('-------------------------------------');
   writeln('  DATE    UNIT        UNIT     UPDATE ');
   writeln('                       NAME     SOURCE ');
   writeln('-------------------------------------');
   writeln;
   writeln('                        (1)       (1)   ');
   writeln('-------------------------------------');
   writeln;writeln;
   for i:= 1 to num_candidates do
     begin
       write(history(.analogy_candidate(.i.).index_num.).date,'   ');
       write(history(.analogy_candidate(.i.).index_num.).unit,'  ');
       if history(.analogy_candidate(.i.).index_num.).unit =
          newrecord.unit then write('YES       ')
       else write(' NO       ');
       if history(.analogy_candidate(.i.).index_num.).update_source =
```

```
          factual then writeln('YES')
       else writeln(' NO');
       writeln
    end;(*end for loop*)
end;(*end procedure print_subsistence reasoning*)


procedure print_fuel_reasoning;
var
  i : integer;                    (*loop control variable              *)
begin
  page;
  writeln;writeln;writeln;writeln;writeln;
  writeln('                    FUEL SUPPLY REASONING');
  writeln;writeln;writeln;writeln;
  writeln('The analogous operations are evaluated on the strength');
  writeln('of their similarity to the current operation in those');
  writeln('areas pertinent to fuel supply consumption. Each of the');
  writeln('points of similarity are weighted independently.');
  writeln;
  writeln('The weighting of each item is in parenthesis below the ');
  writeln('item name. i.e. (2) = 2 points for AF_GROUND_SUPPORT');
  writeln;
  writeln('Up to three previous operations are considered in the ');
  writeln('adjustment algorithm, with those operations with the ');
  writeln('highest number of quality points being chosen for this ');
  writeln('purpose.');
  writeln;writeln;writeln;writeln;writeln;
  write('-------------------------------------------------------');
  writeln('-------------------------');
  write('  DATE    UNIT          AF        TERRAIN      ');
  writeln('UNIT   COUNTRY   OPERATION   UPDATE');
  write('                     GRND_SPT                   ');
  writeln('NAME    NAME        NAME     SOURCE');
  write('-------------------------------------------------------');
  writeln('-------------------------');
  writeln;
  write('                          (2)         (3)         ');
  writeln('  (1)     (1)          (1)        (2)  ');
  write('-------------------------------------------------------');
  writeln('-------------------------');
  writeln;writeln;
  for i:= 1 to num_candidates do
    begin
      write(history(.analogy_candidate(.i.).index_num.).date,'   ');
      write(history(.analogy_candidate(.i.).index_num.).unit,'  ');
      if history(.analogy_candidate(.i.).index_num.).AF_ground_spt =
         newrecord.AF_ground_spt then write('YES       ')
      else write(' NO        ');
      if history(.analogy_candidate(.i.).index_num.).terrain =
         newrecord.terrain then write('YES       ')
      else write(' NO        ');
      if history(.analogy_candidate(.i.).index_num.).unit =
         newrecord.unit then write('  YES     ')
      else write('   NO     ');
      if history(.analogy_candidate(.i.).index_num.).country =
         newrecord.country then write('YES       ')
      else write(' NO        ');
      if history(.analogy_candidate(.i.).index_num.).operation_name
         = newrecord.operation_name then write('  YES     ')
      else write('   NO     ');
      if history(.analogy_candidate(.i.).index_num.).update_source =
         factual then writeln('   YES')
      else writeln('    NO');
      writeln
    end;(*end for loop*)
end;(*end procedure print_fuel_reasoning*)
```

```
procedure print_ammo_reasoning;
var
   i : integer;                    (*loop control variable              *)
begin
   page;
   writeln;writeln;writeln;writeln;writeln;
   writeln('                    AMMUNITION REASONING');
   writeln;writeln;writeln;writeln;
   writeln('The analogous operations are evaluated on the strength');
   writeln('of their similarity to the current operation in those');
   writeln('areas pertinent to ammo supply consumption. Each of the');
   writeln('points of similarity are weighted independently.');
   writeln;
   writeln('The weighting of each item is in parenthesis below the ');
   writeln('item name. i.e. (3) = 3 points for AF_GROUND_SUPPORT');
   writeln;
   writeln('The three previous operations with the highest number of');
   writeln('quality points are used in the adjustment algorithm.');
   writeln;writeln;writeln;writeln;writeln;
   write('-------------------------------------------------------');
   writeln('-----------------------------------');
   write('  DATE     UNIT          AF        TERRAIN   VISIBILITY   ');
   writeln('UNIT   COUNTRY   OPERATION   UPDATE');
   write('                      GRND_SPT                           ');
   writeln('NAME    NAME         NAME     SOURCE');
   write('-------------------------------------------------------');
   writeln('-----------------------------------');
   writeln;
   write('                          (3)         (2)         (1)       ');
   writeln(' (1)     (1)          (1)        (1)  ');
   write('-------------------------------------------------------');
   writeln('-----------------------------------');
   writeln;writeln;
   for i:= 1 to num_candidates do
     begin
       write(history(.analogy_candidate(.i.).index_num.).date,'   ');
       write(history(.analogy_candidate(.i.).index_num.).unit,'  ');
       if history(.analogy_candidate(.i.).index_num.).AF_ground_spt =
          newrecord.AF_ground_spt then write('YES        ')
       else write(' NO        ');
       if history(.analogy_candidate(.i.).index_num.).terrain =
          newrecord.terrain then write('YES        ')
       else write(' NO        ');
       if history(.analogy_candidate(.i.).index_num.).visibility =
          newrecord.visibility then write('  YES      ')
       else write('   NO      ');
       if history(.analogy_candidate(.i.).index_num.).unit =
          newrecord.unit then write('  YES      ')
       else write('   NO      ');
       if history(.analogy_candidate(.i.).index_num.).country =
          newrecord.country then write('YES        ')
       else write(' NO        ');
       if history(.analogy_candidate(.i.).index_num.).operation_name
          = newrecord.operation_name then write('  YES      ')
       else write('   NO      ');
       if history(.analogy_candidate(.i.).index_num.).update_source =
          factual then writeln('   YES')
       else writeln('    NO');
       writeln
     end;(*end for loop*)
end;(*end procedure print_ammo_reasoning*)


procedure print_Gen_supplies_reasoning;
var
   i : integer;                    (*loop control variable              *)
begin
   page;
   writeln;writeln;writeln;writeln;writeln;
```

```
   writeln('                    GENERAL SUPPLY REASONING');
   writeln;writeln;writeln;writeln;
   writeln('The analogous operations are evaluated on the strength');
   writeln('of their similarity to the current operation in those');
   writeln('areas pertinent to general supply consumption. Each of the');
   writeln('points of similarity are weighted independently.');
   writeln;
   writeln('The weighting of each item is in parenthesis below the ');
   writeln('item name. i.e. (2) = 2 points for AF_GROUND_SUPPORT');
   writeln;
   writeln('Up to three previous operations are considered in the ');
   writeln('adjustment algorithm, with those operations with the ');
   writeln('highest number of quality points being chosen for this ');
   writeln('purpose.');
   writeln;writeln;writeln;writeln;writeln;
   write('-------------------------------------------------------');
   writeln('-----------------------------------');
   write('  DATE     UNIT          AF        TERRAIN   VISIBILITY   ');
   writeln('UNIT   COUNTRY   OPERATION   UPDATE');
   write('                     GRND_SPT                            ');
   writeln('NAME     NAME        NAME      SOURCE');
   write('-------------------------------------------------------');
   writeln('-----------------------------------');
   writeln;
   write('                          (2)         (2)         (1)       ');
   writeln(' (1)      (1)           (1)        (2) ');
   write('-------------------------------------------------------');
   writeln('-----------------------------------');
   writeln;writeln;
   for i:= 1 to num_candidates do
     begin
       write(history(.analogy_candidate(.i.).index_num.).date,'   ');
       write(history(.analogy_candidate(.i.).index_num.).unit,'  ');
       if history(.analogy_candidate(.i.).index_num.).AF_ground_spt =
          newrecord.AF_ground_spt then write('YES       ')
       else write(' NO        ');
       if history(.analogy_candidate(.i.).index_num.).terrain =
          newrecord.terrain then write('YES       ')
       else write(' NO        ');
       if history(.analogy_candidate(.i.).index_num.).visibility =
          newrecord.visibility then write('  YES      ')
       else write('   NO      ');
       if history(.analogy_candidate(.i.).index_num.).unit =
          newrecord.unit then write('  YES      ')
       else write('   NO      ');
       if history(.analogy_candidate(.i.).index_num.).country =
          newrecord.country then write('YES       ')
       else write(' NO        ');
       if history(.analogy_candidate(.i.).index_num.).operation_name
          = newrecord.operation_name then write('  YES      ')
       else write('   NO      ');
       if history(.analogy_candidate(.i.).index_num.).update_source =
          factual then writeln('   YES')
       else writeln('    NO');
       writeln
     end;(*end for loop*)
end;(*end procedure print_Gen_supplies_reasoning*)


begin (*procedure adjust_estimate*)
  newrecord.analogy_info.num_analogies:= 0;
  index:= file_counter;
  num_candidates:= 0;
  while (num_candidates < max_candidates) and (index > 0) do
     begin
       if analogous(index) then
           begin
             num_candidates:= num_candidates + 1;
             if newrecord.analogy_info.num_analogies < 3 then
```

```
                newrecord.analogy_info.num_analogies:=
                newrecord.analogy_info.num_analogies + 1;
            analogy_candidate(.num_candidates.).index_num:= index;
          end;
        index:= index -1;
      end;(*end while statement*)
  print_analogies;
  if newrecord.analogy_info.num_analogies > 0 then
      begin
        for i:= 1 to num_candidates do
          begin
            analogy_candidate(.i.).strength_pts:=
                compute_strength(0,0,0,1,1,0,0,0,0,0,1,1,0,0,
                                    analogy_candidate(.i.).index_num);
            analogy_candidate(.i.).used:= false
          end;
        for i:= 1 to newrecord.analogy_info.num_analogies do
          begin
            newrecord.analogy_info.analogies(.i.).analogy_index:=
                pick_best_analogy;
            newrecord.analogy_info.analogies(.i.).date:=
                history(.newrecord.analogy_info.analogies(.i.)
                .analogy_index.).date;
            newrecord.analogy_info.analogies(.i.).unit:=
                history(.newrecord.analogy_info.analogies(.i.)
                .analogy_index.).unit;
            newrecord.analogy_info.analogies(.i.).quality_pts:=
                compute_strength(0,0,0,1,1,0,0,0,0,0,1,1,0,0,
                    newrecord.analogy_info.analogies(.i.).analogy_index)
          end;
        print_water_reasoning;
        adjust(1);
        for i:= 1 to num_candidates do
          begin
            analogy_candidate(.i.).strength_pts:=
                compute_strength(0,0,0,1,0,1,0,0,0,0,1,0,1,0,
                                    analogy_candidate(.i.).index_num);
            analogy_candidate(.i.).used:= false
          end;
        for i:= 1 to newrecord.analogy_info.num_analogies do
          begin
            newrecord.analogy_info.analogies(.i.).analogy_index:=
                pick_best_analogy;
            newrecord.analogy_info.analogies(.i.).date:=
                history(.newrecord.analogy_info.analogies(.i.)
                .analogy_index.).date;
            newrecord.analogy_info.analogies(.i.).unit:=
                history(.newrecord.analogy_info.analogies(.i.)
                .analogy_index.).unit;
            newrecord.analogy_info.analogies(.i.).quality_pts:=
                compute_strength(0,0,0,1,0,1,0,0,0,0,1,0,1,0,
                    newrecord.analogy_info.analogies(.i.).analogy_index)
          end;
        print_subsistence_reasoning;
        adjust(2);
        adjust(3);
        for i:= 1 to num_candidates do
          begin
            analogy_candidate(.i.).strength_pts:=
                compute_strength(1,1,1,1,1,1,1,2,1,2,2,1,1,1,
                                    analogy_candidate(.i.).index_num);
            analogy_candidate(.i.).used:= false
          end;
        for i:= 1 to newrecord.analogy_info.num_analogies do
          begin
            newrecord.analogy_info.analogies(.i.).analogy_index:=
                pick_best_analogy;
            newrecord.analogy_info.analogies(.i.).date:=
                history(.newrecord.analogy_info.analogies(.i.)
                .analogy_index.).date;
```

```
      newrecord.analogy_info.analogies(.i.).unit:=
          history(.newrecord.analogy_info.analogies(.i.)
          .analogy_index.).unit;
      newrecord.analogy_info.analogies(.i.).quality_pts:=
          compute_strength(1,1,1,1,1,1,1,2,1,2,2,1,1,1,
              newrecord.analogy_info.analogies(.i.).analogy_index)
    end;
print_Gen_supplies_reasoning;
adjust(4);
adjust(6);
adjust(19);
adjust(20);
adjust(21);
for i:= 1 to num_candidates do
    begin
      analogy_candidate(.i.).strength_pts:=
          compute_strength(1,0,1,1,1,1,1,2,0,3,1,1,1,1,
                                analogy_candidate(.i.).index_num);
      analogy_candidate(.i.).used:= false
    end;
for i:= 1 to newrecord.analogy_info.num_analogies do
    begin
      newrecord.analogy_info.analogies(.i.).analogy_index:=
          pick_best_analogy;
      newrecord.analogy_info.analogies(.i.).date:=
          history(.newrecord.analogy_info.analogies(.i.)
          .analogy_index.).date;
      newrecord.analogy_info.analogies(.i.).unit:=
          history(.newrecord.analogy_info.analogies(.i.)
          .analogy_index.).unit;
      newrecord.analogy_info.analogies(.i.).quality_pts:=
          compute_strength(1,0,1,1,1,1,1,2,0,3,1,1,1,1,
              newrecord.analogy_info.analogies(.i.).analogy_index)
    end;
print_fuel_reasoning;
adjust(5);
for i:= 1 to num_candidates do
    begin
      analogy_candidate(.i.).strength_pts:=
          compute_strength(1,1,1,1,1,1,1,3,1,2,1,1,1,1,
                                analogy_candidate(.i.).index_num);
      analogy_candidate(.i.).used:= false
    end;
for i:= 1 to newrecord.analogy_info.num_analogies do
    begin
      newrecord.analogy_info.analogies(.i.).analogy_index:=
          pick_best_analogy;
      newrecord.analogy_info.analogies(.i.).date:=
          history(.newrecord.analogy_info.analogies(.i.)
          .analogy_index.).date;
      newrecord.analogy_info.analogies(.i.).unit:=
          history(.newrecord.analogy_info.analogies(.i.)
          .analogy_index.).unit;
      newrecord.analogy_info.analogies(.i.).quality_pts:=
          compute_strength(1,1,1,1,1,1,1,3,1,2,1,1,1,1,
              newrecord.analogy_info.analogies(.i.).analogy_index)
    end;
print_ammo_reasoning;
adjust(7);
adjust(8);
adjust(9);
adjust(10);
adjust(11);
adjust(12);
adjust(13);
adjust(14);
adjust(15);
adjust(16);
adjust(17);
adjust(18)
```

```
      end
   else for adjustment_index:= 1 to num_supply_items do
           newrecord.consumption(.adjustment_index.).adjustments:= 0;
   page;
end; (*end procedure adjust_estimate*)


begin (*begin procedure create_estimate*)
   water_estimate;
   class_I_estimate;
   compute_general_supplies( 4, 3.67);
   compute_general_supplies( 6, 4.00);
   compute_general_supplies(19,15.00);
   compute_general_supplies(20, 1.22);
   compute_general_supplies(21, 2.50);
   diesel_fuel_estimate;
   if newrecord.duration = first_day then
     begin
        compute_ammo( 7,taskforce(.13.).quantity+taskforce(.14.).quantity,
                                                    52,62,29,35,10,12);
        compute_ammo( 8,taskforce(.15.).quantity,7,9,4,6,3,4);
        compute_ammo( 9,taskforce(.16.).quantity,2,3,1,2,1,1);
        compute_ammo(10,taskforce(. 7.).quantity,376,423,244,275,132,148);
        compute_ammo(11,taskforce(. 8.).quantity,374,520,229,318,115,160);
        compute_ammo(12,taskforce(. 9.).quantity,288,395,186,255, 84,115);
        compute_ammo(13,taskforce(.11.).quantity,3984,4800,2241,2700,747,
                                                          900);
        compute_ammo(14,taskforce(. 5.).quantity, 97,116,54,65,18,22);
        compute_ammo(15,taskforce(. 6.).quantity,109,130,61,73,20,24);
        compute_ammo(16,taskforce(.17.).quantity,175,210,99,118,33,39);
        compute_ammo(17,taskforce(.18.).quantity,433,519,243,292,81,97);
        compute_ammo(18,taskforce(.19.).quantity,99,118,56,67,19,22)
     end
   else begin
        compute_ammo( 7,taskforce(.13.).quantity+taskforce(.14.).quantity,
                                                    28,38,16,21,5,7);
        compute_ammo( 8,taskforce(.15.).quantity,8,10,5,7,3,4);
        compute_ammo( 9,taskforce(.16.).quantity,3,4,2,2,1,10);
        compute_ammo(10,taskforce(. 7.).quantity,381,467,248,304,133,163);
        compute_ammo(11,taskforce(. 8.).quantity,374,530,229,324,120,163);
        compute_ammo(12,taskforce(. 9.).quantity,281,363,181,235,82,106);
        compute_ammo(13,taskforce(.11.).quantity,2151,2880,1210,1620,403,
                                                          540);
        compute_ammo(14,taskforce(. 5.).quantity, 53,70,30,40,10,13);
        compute_ammo(15,taskforce(. 6.).quantity,59,79,33,45,11,15);
        compute_ammo(16,taskforce(.17.).quantity,96,127,54,72,18,24);
        compute_ammo(17,taskforce(.18.).quantity,236,314,133,177,44,59);
        compute_ammo(18,taskforce(.19.).quantity,54,72,30,40,10,13)
         end;
   adjust_estimate;
   for i:= 1 to num_supply_items do
      newrecord.consumption(.i.).final_estimate:=
      newrecord.consumption(.i.).general_estimate +
      newrecord.consumption(.i.).adjustments;
   file_counter:= file_counter + 1;
   history(.file_counter.):= newrecord
end; (*end procedure create_estimate*)


procedure print_estimate;
var
   i : integer;          (*index variable for printing consumption array*)
begin
   page;writeln;writeln;
   write('                              ');
   writeln('AUTOMATED LOGISTICS PLAN');
   writeln;writeln;writeln;
   writeln('DATE                    ',newrecord.date);
   writeln('UNIT                    ',newrecord.unit);
```

```
case newrecord.tf_type of
    armor  : writeln('TASK FORCE TYPE        ARMOR');
    mech   : writeln('TASK FORCE TYPE        MECHANIZED');
    inf    : writeln('TASK FORCE TYPE        INFANTRY');
end; (*end case statement*)
case newrecord.tf_size of
    bn     : writeln('TASK FORCE SIZE        BATTALION');
    bde    : writeln('TASK FORCE SIZE        BRIGADE');
end;
case newrecord.mission of
    attack : writeln('MISSION                ATTACK');
    defend : writeln('MISSION                DEFEND');
end;
case newrecord.duration of
    first_day      :  writeln('DURATION                FIRST DAY');
    succeeding_day :  writeln('DURATION                SUCCEEDING DAY');
end;(*end case statement*)
case newrecord.intensity of
    hi     : writeln('COMBAT INTENSITY       HIGH');
    mid    : writeln('COMBAT INTENSITY       MID');
    low    : writeln('COMBAT INTENSITY       LOW');
end; (*end case statement*)
writeln('OPERATION NAME          ',newrecord.operation_name);
case newrecord.area of
    conus  : writeln('AREA                   CONUS');
    europe : writeln('AREA                   EUROPE');
    korea  : writeln('AREA                   KOREA')
end; (*end case statement*)
writeln('COUNTRY                 ',newrecord.country);
case newrecord.climate of
    hot        :  writeln('CLIMATE                HOT');
    temperate  :  writeln('CLIMATE                TEMPERATE');
    cold       :  writeln('CLIMATE                COLD')
end;(*end case statement*)
case newrecord.terrain of
    open     : writeln('TERRAIN              OPEN');
    woods    : writeln('TERRAIN              WOODS');
    built_up : writeln('TERRAIN              BUILT UP');
    mountains: writeln('TERRAIN              MOUNTAINS');
end; (*end case statement*)
case newrecord.visibility of
    good   : writeln('VISIBILTY              GOOD');
    fair   : writeln('VISIBILTY              FAIR');
    poor   : writeln('VISIBILITY             POOR');
end; (*end case statement*)
if newrecord.AF_ground_spt = true then
      writeln('AF GROUND SUPPORT      YES')
else writeln('AF GROUND SUPPORT      NO');
if newrecord.moppcondition = true then
      writeln('MOPP LEVEL 3/4         YES')
else writeln('MOPP LEVEL 3/4         NO');
writeln('PERSONNEL STRENGTH      ',newrecord.personnel_strength:4);
if newrecord.ration_policy = b_c_b then
      writeln('RATION POLICY          b_c_b')
else writeln('RATION POLICY          c_c_b');
writeln;writeln;
write('HISTORICAL DATA AVAILABLE  ');
if newrecord.analogy_info.num_analogies > 0 then
    begin
      writeln('YES');
      writeln;
      write('   DATE   ');
      for i:= 1 to newrecord.analogy_info.num_analogies do
          write(newrecord.analogy_info.analogies(.i.).date,'   ');
      writeln;
      write('   UNIT   ');
      for i:= 1 to newrecord.analogy_info.num_analogies do
          write(newrecord.analogy_info.analogies(.i.).unit,' ');
      writeln;
    end
```

```
   else writeln('NO');
   writeln;writeln;
   write('                                        ');
   writeln('LOGISTICS ESTIMATE');
   writeln;writeln;
   write('  SUPPLY ITEM              GENERAL EST.     ADJUSTMENTS');
   writeln('     FINAL EST.');
   writeln;writeln;
   for i:= 1 to num_supply_items do
       begin
         write(newrecord.consumption(.i.).supply_item);
         write('          ');
         write(newrecord.consumption(.i.).general_estimate:6);
         write('          ');
         write(newrecord.consumption(.i.).adjustments:6);
         write('          ');
         write(newrecord.consumption(.i.).final_estimate:6);
         write('       ');
         writeln(newrecord.consumption(.i.).unit_of_measure);
         writeln
       end; (*end printing out consumption array*)
 end; (*end procedure printestimate*)

begin (*begin log_estimate*)
   build_task_force;
   create_scenario;
   create_estimate;
   print_estimate;
   page
end;  (*end procedure log_estimate*)


(*************************************************************************
              MODULE HISTORY_UPDATE
 *************************************************************************)
procedure history_update;
var
   update_record : oprecord;  (*record to be updated                    *)
   i             : integer;   (*index into the historical files        *)
   found         : boolean;   (*true if record found in history files *)
   continue_char : char;      (*user response to continue with program*)
procedure readunit;
const
   blanks = '       ';
var
   ok       : boolean;
   answer   : char;
   unitname : unitstring;
begin
   ok:= false;
   repeat
      writeln('Enter name of unit which conducted the operation');
      writeln('For example- 1/33rd ');
      readln(unitname);
      strconcat(unitname,blanks);
      writeln;
      writeln('The name of the unit was ', unitname);
      writeln;writeln;
      writeln('Is this the correct unit name?');
      writeln('Enter the number corresponding to your answer.');
      writeln('    1 - yes, unit name is correct  ');
      writeln('    2 - no,  unit name is incorrect');
      readln(answer);
      if answer = '1' then
        begin
          update_record.unit:= unitname;
          ok:= true
        end
```

```
   until ok = true;
   writeln;
end; (*end procedure readunit*)

procedure readdate;
var
   ok      : boolean;
   newdate : datestring;
   answer  : char;
begin
   ok:= false;
   repeat
      writeln('Enter the date on which the operation took place.');
      writeln('Use the form dd/mm/yy');
      readln(newdate);
      writeln;
      writeln('The date of the operation was ', newdate);
      writeln;
      writeln('Is this the correct date?');
      writeln('Enter the number corresponding to your answer.');
      writeln('    1 - yes, date is correct  ');
      writeln('    2 - no,  date is incorrect');
      readln(answer);
      if answer = '1' then
        begin
           update_record.date:= newdate;
           ok:= true
        end
   until ok = true;
   writeln;
end; (*end procedure readdate*)

procedure readmission;
var
   ok : boolean;
   mission_code : char;
begin
   writeln('Enter the number corresponding to the correct mission.');
   writeln('    1 - attack  ');
   writeln('    2 - defend  ');
   ok:= false;
   repeat
       readln(mission_code);
       if mission_code = '1' then
           begin
              update_record.mission := attack;
              ok:= true
           end
       else if mission_code = '2' then
           begin
              update_record.mission:= defend;
              ok:= true
           end
       else writeln('you have made an error in input, try again.');
   until ok = true;
   writeln;
end; (*end procedure readmission*)


procedure readupdate_source;
var
   ok     : boolean;
   answer : char;
begin
   writeln('What was the source of the information for this update');
   writeln;
   writeln('Enter the correct number for your response');
   writeln('   1 - estimate');
```

```
   writeln('   2 - factual information ');
   ok:= false;
   repeat
     readln(answer);
     if answer = '1' then
       begin
         history(.i.).update_source:= estimate;
         ok:= true
       end
     else if answer = '2' then
       begin
         history(.i.).update_source:= factual;
         ok:= true
       end
     else writeln('You have made an error in input, try again.');
   until ok = true;
   writeln;writeln;writeln
end; (*end procedure readupdate_source*)


procedure input_consumption;
var
   j,                    (*index variable into the consumption array     *)
   amount : integer;     (*the user provided consumption figures         *)
begin
   with history(.i.) do
     begin
       writeln('Enter the actual consumption for each of the supply ');
       writeln('items that follow. If no actual consumption figures ');
       writeln('are available, enter 0 .');
       writeln;writeln;
       for j:= 1 to num_supply_items do
         begin
           write('Enter the number of ');
           write(consumption(.j.).supply_item,' ');
           writeln(consumption(.j.).unit_of_measure,'     ');
           readln(amount);
           writeln(amount);
           consumption(.j.).actual_consumption:= amount;
           writeln
         end;
       update:= true
     end
end; (*end procedure input_consumption*)


begin (*begin history_update*)
   writeln('You will now be asked information about the operation.');
   writeln('for which you have actual consumption data.');writeln;
   readunit;
   readdate;
   readmission;
   found:= false;
   i:= 1;
   while (not found) and (not (i > file_counter)) do
      with update_record do
        begin
          if (history(.i.).unit    = unit) and
             (history(.i.).date    = date) and
             (history(.i.).mission = mission)
          then begin
                 found:= true;
                 readupdate_source;
                 input_consumption
               end
          else i:= i+ 1
        end;
   if found then begin
      writeln;writeln;
```

```
      writeln('The record has been updated .')
      end
   else
      begin
         writeln('There is no record in the historical file which ');
         writeln('matches the unit,date, and mission you have specified');
         writeln('Check your input and try again ')
      end;
   writeln('Enter c to continue');
   repeat
     readln(continue_char)
   until continue_char = 'c';
   page
end; (*end procedure history_update*)


(*************************************************************************
                          MODULE DELETE_RECORD
*************************************************************************)
procedure delete_record;
var
   delete_record : oprecord;  (*name of record to be deleted          *)
   i             : integer;   (*index into the historical files       *)
   found         : boolean;   (*true if record found in history files *)
   continue_char : char;      (*user response to continue program     *)
procedure readunit;
const
   blanks = '       ';
var
   ok       : boolean;
   answer   : char;
   unitname : unitstring;
begin
   ok:= false;
   repeat
     writeln('Enter name of unit which conducted the operation');
     writeln('For example- 1/33rd ');
     readln(unitname);
     strconcat(unitname,blanks);
     writeln;
     writeln('The name of the unit was ', unitname);
     writeln;writeln;
     writeln('Is this the correct unit name?');
     writeln('Enter the number corresponding to your answer.');
     writeln('    1 - yes, unit name is correct  ');
     writeln('    2 - no,  unit name is incorrect');
     readln(answer);
     if answer = '1' then
       begin
          delete_record.unit:= unitname;
          ok:= true
       end
   until ok = true;
   writeln
end; (*end procedure readunit*)

procedure readdate;
var
   ok      : boolean;
   newdate : datestring;
   answer  : char;
begin
   ok:= false;
   repeat
     writeln('Enter the date on which the operation took place.');
     writeln('Use the form dd/mm/yy');
     readln(newdate);
     writeln;
```

```
      writeln('The date of the operation was ', newdate);
      writeln;
      writeln('Is this the correct date?');
      writeln('Enter the number corresponding to your answer.');
      writeln('    1 - yes, date is correct  ');
      writeln('    2 - no,  date is incorrect');
      readln(answer);
      if answer = '1' then
        begin
          delete_record.date:= newdate;
          ok:= true
        end
    until ok = true;
    writeln
end; (*end procedure readdate*)

procedure readmission;
var
    ok : boolean;
    mission_code : char;
begin
    ok:= false;
    repeat
        writeln('Enter the number corresponding to the correct mission.');
        writeln('    1 - attack  ');
        writeln('    2 - defend  ');
        readln(mission_code);
        if mission_code = '1' then
            begin
              delete_record.mission := attack;
              ok:= true
            end
        else if mission_code = '2' then
            begin
              delete_record.mission:= defend;
              ok:= true
            end
        else writeln('you have made an error in input, try again.');
    until ok = true;
    writeln
end; (*end procedure readmission*)


procedure deletion;
var
    j : integer;                (*index variable into the history files*)
begin
    for j:= i to (file_counter -1) do
        history(.j.):= history(.j+1.);
    file_counter:= file_counter -1;
    writeln;writeln;
    writeln('The record was found and deleted.')
end; (*end procedure deletion*)


begin (*begin module delete_record*)
    writeln('You will now be asked information about the operation.');
    writeln('that you want deleted.');
    writeln;
    readunit;
    readdate;
    readmission;
    found:= false;
    i:= 1;
    while (not found) and (not (i > file_counter)) do
        with delete_record do
          begin
            if (history(.i.).unit    = unit) and
```

```
            (history(.i.).date    = date) and
            (history(.i.).mission = mission)
          then begin
                 found:= true;
                 deletion
               end
          else i:= i+ 1
        end;
   if (not found) then
      begin
        writeln('There is no record in the historical file which ');
        writeln('matches the unit,date, and mission you have specified');
        writeln('Check your input and try again ')
      end;
   writeln;writeln;writeln;write('                    ');
   writeln('Enter c to continue');
   repeat
     readln(continue_char)
   until continue_char = 'c';
   page
end; (*end procedure delete_record*)


(*************************************************************************
                          MODULE PRINT_HISTORY
*************************************************************************)
procedure print_history;
var
   j,                    (*index variable for history array            *)
   i : integer;          (*index variable for printing consumption array*)
begin
   for j:= 1 to file_counter do
     begin
        writeln;writeln;
        write('                                  ');
        writeln('HISTORICAL RECORD');
        writeln;writeln;writeln;
        writeln('DATE                    ',history(.j.).date);
        writeln('UNIT                    ',history(.j.).unit);
        case history(.j.).tf_type of
            armor  : writeln('TASK FORCE TYPE        ARMOR');
            mech   : writeln('TASK FORCE TYPE        MECHANIZED');
            inf    : writeln('TASK FORCE TYPE        INFANTRY');
        end; (*end case statement*)
        case history(.j.).tf_size of
            bn     : writeln('TASK FORCE SIZE        BATTALION');
            bde    : writeln('TASK FORCE SIZE        BRIGADE');
        end; (*end case statement*)
        if history(.j.).mission = attack then
             writeln('MISSION                ATTACK')
        else writeln('MISSION               DEFEND');
        case history(.j.).duration of
          first_day     :writeln('DURATION                FIRST DAY');
          succeeding_day:writeln('DURATION                SUCCEEDING DAY');
        end;(*end case statement*)
        case history(.j.).intensity of
            hi     : writeln('COMBAT INTENSITY       HIGH');
            mid    : writeln('COMBAT INTENSITY       MID');
            low    : writeln('COMBAT INTENSITY       LOW');
        end; (*end case statement*)
        writeln('OPERATION NAME          ',history(.j.).operation_name);
        case history(.j.).area of
            conus  : writeln('AREA                    CONUS');
            europe : writeln('AREA                    EUROPE');
            korea  : writeln('AREA                    KOREA')
        end; (*end case statement*)
        writeln('COUNTRY                 ',history(.j.).country);
        case history(.j.).climate of
            hot       :  writeln('CLIMATE                HOT');
```

```
    temperate :  writeln('CLIMATE                TEMPERATE');
    cold      :  writeln('CLIMATE                COLD')
end;(*end case statement*)
case history(.j.).terrain of
    open     : writeln('TERRAIN                 OPEN');
    woods    : writeln('TERRAIN                 WOODS');
    built_up : writeln('TERRAIN                 BUILT UP');
    mountains: writeln('TERRAIN                 MOUNTAINS');
end; (*end case statement*)
case history(.j.).visibility of
    good   : writeln('VISIBILTY              GOOD');
    fair   : writeln('VISIBILTY              FAIR');
    poor   : writeln('VISIBILITY             POOR');
end; (*end case statement*)
if history(.j.).AF_ground_spt = true then
      writeln('AF GROUND SUPPORT      YES')
else writeln('AF GROUND SUPPORT      NO');
if history(.j.).moppcondition = true then
      writeln('MOPP LEVEL 3/4         YES')
else writeln('MOPP LEVEL 3/4         NO');
write('PERSONNEL STRENGTH     ');
writeln(history(.j.).personnel_strength:4);
if history(.j.).ration_policy = b_c_b then
   writeln('RATION POLICY          b_c_b')
else writeln('RATION POLICY          c_c_b');
if (history(.j.).update = true) then
    case history(.j.).update_source of
        none      : writeln('UPDATE SOURCE          NONE');
        estimate  : writeln('UPDATE SOURCE          ESTIMATE');
        factual   : writeln('UPDATE SOURCE          FACTUAL');
    end (*end case statement*)
else writeln('UPDATE SOURCE          NONE');
writeln;writeln;
write('HISTORICAL DATA AVAILABLE  ');
if history(.j.).analogy_info.num_analogies > 0 then
    begin
      writeln('YES');
      writeln;
      write('   DATE   ');
      for i:= 1 to history(.j.).analogy_info.num_analogies do
        begin
          write(history(.j.).analogy_info.analogies(.i.).date);
          write('    ')
        end;
      writeln;
      write('   UNIT   ');
      for i:= 1 to history(.j.).analogy_info.num_analogies do
        write(history(.j.).analogy_info.analogies(.i.).unit,'   ');
      writeln;
    end
else writeln('NO');
writeln;writeln;
write('                                    ');
writeln('LOGISTICS ESTIMATE');
writeln;writeln;
write('  SUPPLY ITEM           GENERAL EST.     ADJUSTMENTS');
writeln('      FINAL EST.    ACTUAL CONS.');
writeln;writeln;
for i:= 1 to num_supply_items do
  begin
     write(history(.j.).consumption(.i.).supply_item);
     write('          ');
     write(history(.j.).consumption(.i.).general_estimate:6);
     write('          ');
     write(history(.j.).consumption(.i.).adjustments:6);
     write('          ');
     write(history(.j.).consumption(.i.).final_estimate:6);
     write('          ');
     write(history(.j.).consumption(.i.).actual_consumption:6);
     write('          ');
```

```
            writeln(history(.j.).consumption(.i.).unit_of_measure:6);
            writeln
          end; (*end printing out consumption array*)
      page
    end  (*end printing all the files in the history array*)
end;   (*end procedure print_history*)


(*************************************************************************
                          MODULE PRINT_DIRECTORY
 ************************************************************************)
procedure print_directory;
var
   history_count : integer; (*index variable into hitory array          *)
   continue_char : char;    (*user response to continue program        *)
begin
   writeln;writeln;
   write('                                    ');
   writeln('DIRECTORY');
   writeln;writeln;writeln;
   if file_counter = 0 then writeln('There are no files in storage.')
   else begin
          write('          DATE          UNIT          MISSION');
          writeln('          UPDATED');
          writeln;
          for history_count:= 1 to file_counter do
            begin
              write('          ',history(.history_count.).date);
              write('      ',history(.history_count.).unit);
              if history(.history_count.).mission = attack then
                    write('      ATTACK')
              else write('      DEFEND');
              if history(.history_count.).update = true then
                    writeln('             YES')
              else writeln('             NO');
              writeln
            end
          end;
   writeln;
   write('                  ');
   writeln('Enter c to continue');
   repeat
     readln(continue_char)
   until continue_char = 'c';
   page;
end;(*end procedure print_directory*)


(*************************************************************************
                          MODULE END_SESSION
*************************************************************************)
procedure end_session;
var
   i : integer;    (*index variable to write historical files to
                    secondary memory                                    *)
begin
   finished:= true;
   rewrite(history_file,'hist oprecord a');
   for i:= 1 to file_counter do
     write(history_file,history(.i.));
   writeln;writeln;
   writeln('This session is now over.');
   writeln;writeln;
   writeln('The key to modern warfare is logistics!')
end; (*end procedure end_session*)
```

```
(*************************************************************************
                                MAIN PROGRAM
*************************************************************************)

begin (* main program *)
   initialize;
   repeat
     module_choice;
     case module_code of
         '1' : log_estimate;
         '2' : history_update;
         '3' : delete_record;
         '4' : print_history;
         '5' : print_directory;
         '6' : end_session
     end; (*end case statement*)
   until finished = true
end. (*end main program*)
```

APPENDIX E
PARTIAL PROGRAM IMPLEMENTATION IN COMMON LISP

This program is a partial implementation of the automated-logistics-planning system in Appendix D. Specifically, this program performs the referencing and calculations necessary to create the general estimates for the same supply items identified in the automated logistics plans in Appendices A and B. The driver of the program is function **try.** The principal data structures of the program are the user-defined structures: *operation*, *task force*, and *supply-item.* The program accepts input data on task force composition and operation attributes in the same manner as the Pascal implementation of the program. There is no error checking done of user input. Function create-supply-item performs the referencing and calculating involved in creating estimates in accordance with current Army doctrine. The program stops here. Two output documents are produced by the program:

1. Task Force Composition
2. Automated Logistics Plan

These documents are almost identical to their counterparts in appendices A and B. The program does not permanently store information about the estimates it creates nor does it conduct any of the reasoning discussed in chapter 3. One of the interesting features of user-defined structures in Common Lisp is that after a structure has been defined, Common Lisp provides functions that insert and retrieve data from fields within instances of the defined structure. **Make- < structure name >** is such a function. It creates an instance of a structure. The format of this function results in code that is easy to read and understand. Specifically, make- < structure name > requires the programmer to place the value for the fields of the instance of the structure next to the corresponding field names. The program follows.

```
(defun try ()
     (format t  "  ")
     (create-operation)
     (format t "  ")
     (terpri)(terpri)(terpri)(terpri)
     (create-taskforce)
     (create-supply-item)
     (format t "  ")
     (write-output)
     (logistics-output)
     (format t "  ")
     (taskforce-output))



(defstruct operation
             date
             unit
             mission
             climate
             area
             tf-type
             tf-size
             intensity
             moppcondition
             personnel
             ration-policy)



(defun create-operation ()
     (setq op1 (make-operation :date          (read-date)
                               :unit          (read-unit)
                               :mission       (read-mission)
                               :climate       (read-climate)
                               :area          (read-area)
                               :tf-type       (read-tf-type)
                               :tf-size       (read-tf-size)
                               :intensity     (read-intensity)
                               :moppcondition (read-moppcondition)
                               :personnel     (read-personnel)
                               :ration-policy (read-rations))))



(defun read-date ()
    (terpri)(terpri)(terpri)(terpri)(terpri)
    (princ "ENTER THE DATE OF THE OPERATION.         ")
    (read))


(defun read-unit ()
    (terpri)
    (princ "ENTER THE NAME OF THE UNIT.              ")
    (read))
```

```
(defun read-mission ()
    (terpri)
    (princ "ENTER THE MISSION TO BE PERFORMED.       ")
    (read))


(defun read-climate ()
    (terpri)
    (princ "ENTER THE CLIMATE IN WHICH THE           ")
    (terpri)
    (princ "      OPERATION WILL CONDUCTED.          ")
    (read))


(defun read-area ()
    (terpri)
    (princ "ENTER THE AREA OF THE WORLD IN WHICH     ")
    (terpri)
    (princ "      THE OPERATION WILL TAKE PLACE.     ")
    (read))


(defun read-tf-type ()
    (terpri)
    (princ "ENTER THE TYPE OF TASK FORCE CONDUCTING  ")
    (terpri)
    (princ "      THE OPERATION.                     ")
    (read))


(defun read-tf-size ()
    (terpri)
    (princ "ENTER THE SIZE OF THE TASK FORCE.        ")
    (read))


(defun read-intensity ()
    (terpri)
    (princ "ENTER THE INTENSITY EXPECTED.            ")
    (read))


(defun read-moppcondition ()
    (terpri)
    (princ "DO YOU EXPECT THE TASK FORCE TO BE IN    ")
    (terpri)
    (princ "      MOPP LEVEL 3/4 ?                   ")
    (terpri)
    (princ "      ENTER YES OR NO                    ")
    (read))


(defun read-personnel ()
    (terpri)
    (princ "ENTER THE TOTAL NUMBER OF PERSONNEL      ")
    (terpri)
    (princ "      IN THE TASK FORCE                  ")
    (read))


(defun read-rations ()
    (terpri)
    (princ "ENTER THE RATION POLICY DURING THE       ")
    (terpri)
    (princ "      OPERATION. EITHER b-c-b or c-c-b   ")
    (read))
```

```
(defstruct taskforce
            m2
            m3
            m113
            m901
            m125a1
            m106a1
            m102
            m109
            m110
            m1rs
            m163
            m730
            m1
            m60
            tows
            m222
            m2mg
            m60mg
            m16a1)



(defun create-taskforce ()
   (setq tf1 (make-taskforce :m2       (read-data "m2    ")
                             :m3       (read-data "m3    ")
                             :m113     (read-data "m113  ")
                             :m901     (read-data "m901  ")
                             :m125a1   (read-data "m125a1")
                             :m106a1   (read-data "m106a1")
                             :m102     (read-data "m102  ")
                             :m109     (read-data "m109  ")
                             :m110     (read-data "m110  ")
                             :m1rs     (read-data "m1rs  ")
                             :m163     (read-data "m163  ")
                             :m730     (read-data "m730  ")
                             :m1       (read-data "m1    ")
                             :m60      (read-data "m60   ")
                             :tows     (read-data "tows  ")
                             :m222     (read-data "m222  ")
                             :m2mg     (read-data "m2mg  ")
                             :m60mg    (read-data "m60mg ")
                             :m16a1    (read-data "m16a1 "))))



(defun read-data (x)
   (terpri)
   (princ "Enter the number of ")
   (princ x)
   (princ " in the task force  ")
   (read))
```

```
(defstruct supply-item
            water
            b-rations
            mre-rations
            class-II-supplies
            diesel-fuel
            class-IV-supplies
            class-VII-supplies
            class-VIII-supplies
            class-IX-supplies
            tank-ammo-105mm
            tow-ammo
            dragon-ammo
            howitzer-ammo-105mm
            howitzer-ammo-155mm
            howitzer-ammo-8in
            vulcan-ammo-20mm
            mortar-ammo-81mm
            mortar-ammo-107mm
            mg-ammo-.50-caliber
            mg-ammo-7.62mm
            rifle-ammo-5.56mm)
```

```
(defun create-supply-item ()
       (setq supl
         (make-supply-item
                :water                (water-est)
                :b-rations            (b-rats)
                :mre-rations          (mre-rats)
                :class-II-supplies    (compute-factor 3.67)
                :diesel-fuel          (diesel)
                :class-IV-supplies    (compute-factor 4)
                :class-VII-supplies   (compute-factor 15)
                :class-VIII-supplies  (compute-factor 1.22)
                :class-IX-supplies    (compute-factor 2.50)
                :tank-ammo-105mm      (compute-ammo (+(taskforce-m1 tf1)
                                                      (taskforce-m2 tf1))
                                                       52 62 29 35 10 12
                :tow-ammo             (compute-ammo (taskforce-tows tf1)
                                                       7 9 4 6 2 4 )
                :dragon-ammo          (compute-ammo (taskforce-m222 tf1)
                                                       2 3 1 2 1 1 )
                :howitzer-ammo-105mm  (compute-ammo (taskforce-m102 tf1)
                                                    376 423 244 275 132 8
                :howitzer-ammo-155mm  (compute-ammo (taskforce-m109 tf1)
                                                    146 203 95 132 51
                :howitzer-ammo-8in    (compute-ammo (taskforce-m110 tf1)
                                                    130 177 85 115 46
                :vulcan-ammo-20mm     (compute-ammo (taskforce-m163 tf1)
                                                   3984 4800 2241 2700 1
                :mortar-ammo-81mm     (compute-ammo (taskforce-m125a1 tf1)
                                                    97 116 54 65 18 22
                :mortar-ammo-107mm    (compute-ammo (taskforce-m106a1 tf1)
                                                    109 130 61 73 20 2
                :mg-ammo-.50-caliber  (compute-ammo (taskforce-m2mg tf1)
                                                    175 210 99 118 33
                :mg-ammo-7.62mm       (compute-ammo (taskforce-m60mg tf1)
                                                    433 519 243 292 81
                :rifle-ammo-5.56mm    (compute-ammo (taskforce-m16a1 tf1)
                                                    99 118 56 67 19 22
```

```
(defun water-est ()
   (cond ((and (equal (operation-climate op1) 'hot)
               (equal (operation-moppcondition op1) 'yes)
               (equal (operation-ration-policy op1) 'b-c-b))
               (* (operation-personnel op1) 5.4))
         ((and (equal (operation-climate op1) 'hot)
               (equal (operation-moppcondition op1) 'yes)
               (equal (operation-ration-policy op1) 'c-c-b))
               (* (operation-personnel op1) 4.9))
         ((and (equal (operation-climate op1) 'hot)
               (equal (operation-moppcondition op1) 'no)
               (equal (operation-ration-policy op1) 'b-c-b))
               (* (operation-personnel op1) 4.9))
         ((and (equal (operation-climate op1) 'hot)
               (equal (operation-moppcondition op1) 'no)
               (equal (operation-ration-policy op1) 'c-c-b))
               (* (operation-personnel op1) 4.4))
         ((and (equal (operation-climate op1) 'temperate)
               (equal (operation-moppcondition op1) 'yes)
               (equal (operation-ration-policy op1) 'b-c-b))
               (* (operation-personnel op1) 4.7))
         ((and (equal (operation-climate op1) 'temperate)
               (equal (operation-moppcondition op1) 'yes)
               (equal (operation-ration-policy op1) 'c-c-b))
               (* (operation-personnel op1) 4.2))
         ((and (equal (operation-climate op1) 'temperate)
               (equal (operation-moppcondition op1) 'no)
               (equal (operation-ration-policy op1) 'b-c-b))
               (* (operation-personnel op1) 3.2))
         ((and (equal (operation-climate op1) 'temperate)
               (equal (operation-moppcondition op1) 'no)
               (equal (operation-ration-policy op1) 'c-c-b))
               (* (operation-personnel op1) 2.7))
         ((and (equal (operation-climate op1) 'cold)
               (equal (operation-moppcondition op1) 'yes)
               (equal (operation-ration-policy op1) 'b-c-b))
               (* (operation-personnel op1) 3.7))
         ((and (equal (operation-climate op1) 'cold)
               (equal (operation-moppcondition op1) 'yes)
               (equal (operation-ration-policy op1) 'c-c-b))
               (* (operation-personnel op1) 3.2))
         ((and (equal (operation-climate op1) 'cold)
               (equal (operation-moppcondition op1) 'no)
               (equal (operation-ration-policy op1) 'b-c-b))
               (* (operation-personnel op1) 3.7))
         ((and (equal (operation-climate op1) 'cold)
               (equal (operation-moppcondition op1) 'no)
               (equal (operation-ration-policy op1) 'c-c-b))
               (* (operation-personnel op1) 3.2))))


(defun b-rats ()
        (cond ((equal (operation-ration-policy op1) 'b-c-b)
                (* (operation-personnel op1) 2))
              (t  (operation-personnel op1))))


(defun mre-rats()
        (cond ((equal (operation-ration-policy op1) 'c-c-b)
                (* (operation-personnel op1) 2))
              (t  (operation-personnel op1))))


(defun compute-factor (x)
        (/ (* (operation-personnel op1) x ) 2000))
```

```
(defun diesel ()
       (cond ((equal (operation-area op1) 'korea)
              (+ (* (taskforce-m2      tf1)   3.0   6.4)
                 (* (taskforce-m2      tf1)   5.5  18.0)
                 (* (taskforce-m2      tf1)   5.5   8.6)
                 (* (taskforce-m3      tf1)   3.0   6.4)
                 (* (taskforce-m3      tf1)   5.5  18.0)
                 (* (taskforce-m3      tf1)   5.5   8.6)
                 (* (taskforce-m113    tf1)   3.1   1.0)
                 (* (taskforce-m113    tf1)   5.5   8.6)
                 (* (taskforce-m113    tf1)   5.5  10.3)
                 (* (taskforce-m901    tf1)   3.0   1.0)
                 (* (taskforce-m901    tf1)   5.5   8.6)
                 (* (taskforce-m901    tf1)   5.5   8.9)
                 (* (taskforce-m125a1  tf1)   4.1   1.0)
                 (* (taskforce-m125a1  tf1)   5.0   8.6)
                 (* (taskforce-m125a1  tf1)   5.0  10.3)
                 (* (taskforce-m106a1  tf1)   4.1   1.0)
                 (* (taskforce-m106a1  tf1)   5.0  10.0)
                 (* (taskforce-m106a1  tf1)   5.0  13.3)
                 (* (taskforce-m102    tf1)  24.0   0.2)
                 (* (taskforce-m109    tf1)   4.1   1.0)
                 (* (taskforce-m109    tf1)   6.0  11.8)
                 (* (taskforce-m109    tf1)   5.5  16.1)
                 (* (taskforce-m110    tf1)   6.5   1.6)
                 (* (taskforce-m110    tf1)   3.6  12.5)
                 (* (taskforce-m110    tf1)   6.0  14.3)
                 (* (taskforce-m1rs    tf1)   5.0   1.0)
                 (* (taskforce-m1rs    tf1)   4.0   6.2)
                 (* (taskforce-m1rs    tf1)   4.5   8.9)
                 (* (taskforce-m163    tf1)   4.1   1.0)
                 (* (taskforce-m163    tf1)   6.0   5.2)
                 (* (taskforce-m163    tf1)   5.5  13.0)
                 (* (taskforce-m730    tf1)   4.0   0.5)
                 (* (taskforce-m730    tf1)   6.0   1.3)
                 (* (taskforce-m730    tf1)   5.5   2.6)
                 (* (taskforce-m1      tf1)   5.2  10.8)
                 (* (taskforce-m1      tf1)   3.3  56.6)
                 (* (taskforce-m1      tf1)   3.4  44.7)
                 (* (taskforce-m60     tf1)   4.6   2.0)
                 (* (taskforce-m60     tf1)   6.5  28.1)
                 (* (taskforce-m60     tf1)   4.6  35.7)))
             ((equal (operation-area op1) 'europe)
              (+(* (taskforce-m2      tf1)   3.0   6.4)
                 (* (taskforce-m2      tf1)   5.5  18.0)
                 (* (taskforce-m2      tf1)   5.5   8.6)
                 (* (taskforce-m3      tf1)   3.0   6.4)
                 (* (taskforce-m3      tf1)   5.5  18.0)
                 (* (taskforce-m3      tf1)   5.5   8.6)
                 (* (taskforce-m113    tf1)   3.0   1.0)
                 (* (taskforce-m113    tf1)   5.5   8.6)
                 (* (taskforce-m113    tf1)   5.5  10.3)
                 (* (taskforce-m901    tf1)   3.0   1.0)
                 (* (taskforce-m901    tf1)   5.5   8.6)
                 (* (taskforce-m901    tf1)   5.5   8.9)
                 (* (taskforce-m125a1  tf1)   4.0   1.0)
                 (* (taskforce-m125a1  tf1)   5.0   8.6)
                 (* (taskforce-m125a1  tf1)   5.0  10.3)
                 (* (taskforce-m106a1  tf1)   4.0   1.0)
                 (* (taskforce-m106a1  tf1)   5.0  10.0)
                 (* (taskforce-m106a1  tf1)   5.0  13.3)
                 (* (taskforce-m102    tf1)  24.0   0.2)
                 (* (taskforce-m109    tf1)   4.0   1.0)
                 (* (taskforce-m109    tf1)   6.0  11.8)
                 (* (taskforce-m109    tf1)   5.5  16.1)
                 (* (taskforce-m110    tf1)   4.0   1.6)
                 (* (taskforce-m110    tf1)   6.0  12.5)
                 (* (taskforce-m110    tf1)   5.5  14.3)
```

```
                (* (taskforce-m1rs    tf1)   5.0   1.0)
                (* (taskforce-m1rs    tf1)   5.0   6.2)
                (* (taskforce-m1rs    tf1)   4.5   8.9)
                (* (taskforce-m163    tf1)   4.0   1.0)
                (* (taskforce-m163    tf1)   6.0   5.2)
                (* (taskforce-m163    tf1)   5.5  13.0)
                (* (taskforce-m730    tf1)   4.0   0.5)
                (* (taskforce-m730    tf1)   6.0   1.3)
                (* (taskforce-m730    tf1)   5.5   2.6)
                (* (taskforce-m1      tf1)   5.0  10.8)
                (* (taskforce-m1      tf1)   6.5  56.6)
                (* (taskforce-m1      tf1)   5.0  44.7)
                (* (taskforce-m60     tf1)   4.5   2.0)
                (* (taskforce-m60     tf1)   6.5  28.1)
                (* (taskforce-m60     tf1)   4.5  35.7)))
             ((equal (operation-area op1) 'conus)
             (+(* (taskforce-m2     tf1)  3.0   6.4)
                (* (taskforce-m2     tf1)  5.5  18.0)
                (* (taskforce-m2     tf1)  5.5   8.6)
                (* (taskforce-m3     tf1)  3.0   6.4)
                (* (taskforce-m3     tf1)  5.5  18.0)
                (* (taskforce-m3     tf1)  5.5   8.6)
                (* (taskforce-m113   tf1)  7.0   1.0)
                (* (taskforce-m113   tf1)  6.8   8.6)
                (* (taskforce-m113   tf1)  1.9  10.3)
                (* (taskforce-m901   tf1)  3.0   1.0)
                (* (taskforce-m901   tf1)  5.5   8.6)
                (* (taskforce-m901   tf1)  5.5   8.9)
                (* (taskforce-m125a1 tf1)  5.0   1.0)
                (* (taskforce-m125a1 tf1)  3.8   8.6)
                (* (taskforce-m125a1 tf1)  1.6  10.3)
                (* (taskforce-m106a1 tf1)  5.3   1.0)
                (* (taskforce-m106a1 tf1)  3.1  10.0)
                (* (taskforce-m106a1 tf1)  4.3  13.3)
                (* (taskforce-m102   tf1) 24.0   0.2)
                (* (taskforce-m109   tf1)  6.2   1.0)
                (* (taskforce-m109   tf1)  1.9  11.8)
                (* (taskforce-m109   tf1)  2.9  16.1)
                (* (taskforce-m110   tf1)  4.1   1.6)
                (* (taskforce-m110   tf1)  1.9  12.5)
                (* (taskforce-m110   tf1)  4.1  14.3)
                (* (taskforce-m1rs   tf1)  5.0   1.0)
                (* (taskforce-m1rs   tf1)  4.0   6.2)
                (* (taskforce-m1rs   tf1)  4.5   8.9)
                (* (taskforce-m163   tf1)  2.4   1.0)
                (* (taskforce-m163   tf1)  7.2   5.2)
                (* (taskforce-m163   tf1)  4.8  13.0)
                (* (taskforce-m730   tf1)  4.0   0.5)
                (* (taskforce-m730   tf1)  6.0   1.3)
                (* (taskforce-m730   tf1)  5.5   2.6)
                (* (taskforce-m1     tf1)  5.2  10.8)
                (* (taskforce-m1     tf1)  3.3  56.6)
                (* (taskforce-m1     tf1)  3.4  44.7)
                (* (taskforce-m60    tf1)  4.2   2.0)
                (* (taskforce-m60    tf1)  8.5  28.1)
                (* (taskforce-m60    tf1)  2.9  35.7)))))
```

```
(defun compute-ammo ( x ha hd ma md la ld )
   (cond ((and (equal (operation-mission op1) 'attack)
               (equal (operation-intensity op1) 'hi))
               (* x ha))
         ((and (equal (operation-mission op1) 'defend)
               (equal (operation-intensity op1) 'hi))
               (* x hd))
```

```
          ((and (equal (operation-mission op1) 'attack)
                (equal (operation-intensity op1) 'mid))
                (* x ma))
          ((and (equal (operation-mission op1) 'defend)
                (equal (operation-intensity op1) 'mid))
                (* x md))
          ((and (equal (operation-mission op1) 'attack)
                (equal (operation-intensity op1) 'low))
                (* x la))
          ((and (equal (operation-mission op1) 'defend)
                (equal (operation-intensity op1) 'low))
                (* x ld))))
```

```
(defun write-output ()
     (terpri)(terpri)(terpri)(terpri)
     (princ "            OPERATION")
     (terpri)
     (princ "   date                 ")
     (prin1 (operation-date op1))
     (terpri)
     (princ "   unit                 ")
     (prin1 (operation-unit op1))
     (terpri)
     (princ "   mission              ")
     (prin1 (operation-mission op1))
     (terpri)
     (princ "   climate              ")
     (prin1 (operation-climate op1))
     (terpri)
     (princ "   area                 ")
     (prin1 (operation-area op1))
     (terpri)
     (princ "   tf-type              ")
     (prin1 (operation-tf-type op1))
     (terpri)
     (princ "   tf-size              ")
     (prin1 (operation-tf-size op1))
     (terpri)
     (princ "   intensity            ")
     (prin1 (operation-intensity op1))
     (terpri)
     (princ "   moppcondition        ")
     (prin1 (operation-moppcondition op1))
     (terpri)
     (princ "   personnel-strength   ")
     (prin1 (operation-personnel op1))
     (terpri)
     (princ "   ration-policy        ")
     (prin1 (operation-ration-policy op1))
     (terpri))
```

```
(defun taskforce-output ()
    (terpri)(terpri)(terpri)(terpri)(terpri)
    (princ "          TASKFORCE COMPOSITION")
    (terpri)(terpri)(terpri)
    (princ " M2 INF FIGHTING VEHICLE      ")
    (prin1 (taskforce-m2 tf1))
    (terpri)
    (princ " M3 CAV FIGHTING VEHICLE      ")
    (prin1 (taskforce-m3 tf1))
    (terpri)
    (princ " M113 PERS CARRIER            ")
    (prin1 (taskforce-m113 tf1))
    (terpri)
    (princ " M901 CBT VEH ITV             ")
    (prin1 (taskforce-m901 tf1))
    (terpri)
    (princ " M125A1 81MM CARRIER          ")
    (prin1 (taskforce-m125a1 tf1))
    (terpri)
    (princ " M106A1 107MM CARRIER         ")
    (prin1 (taskforce-m106a1 tf1))
    (terpri)
    (princ " M102 105MM HOWITZER          ")
    (prin1 (taskforce-m102 tf1))
    (terpri)
    (princ " M109 155MM SP HOWITZER       ")
    (prin1 (taskforce-m109 tf1 ))
    (terpri)
    (princ " M110 8INCH SP HOWITZER       ")
    (prin1 (taskforce-m110 tf1 ))
    (terpri)
    (princ " LAUNCH-LOAD MLRS             ")
    (prin1 (taskforce-mlrs tf1 ))
    (terpri)
    (princ " M163 VULCAN AIR DEFENSE      ")
    (prin1 (taskforce-m163 tf1 ))
    (terpri)
    (princ " M730 CHAP AIR DEFENSE        ")
    (prin1 (taskforce-m730 tf1))
    (terpri)
    (princ " M1 TANK 105MM                ")
    (prin1 (taskforce-m1 tf1 ))
    (terpri)
    (princ " M60 TANK 105MM               ")
    (prin1 (taskforce-m60 tf1))
    (terpri)
    (princ " TOW LAUNCHER                 ")
    (prin1 (taskforce-tows tf1))
    (terpri)
    (princ " M222 DRAGON LAUNCHER         ")
    (prin1 (taskforce-m222 tf1))
    (terpri)
    (princ " M2 50 CALIBER MG             ")
    (prin1 (taskforce-m2mg tf1))
    (terpri)
    (princ " M60 MG                       ")
    (prin1 (taskforce-m60mg tf1))
    (terpri)
    (princ " M16A1 RIFLE                  ")
    (prin1 (taskforce-m16a1 tf1 ))
    (format t "  "))
```

```
(defun logistics-output ()
     (terpri)(terpri)
     (princ "                LOGISTICS  ESTIMATE ")
     (terpri)(terpri)
     (princ "  SUPPLY ITEM                          GENERAL ESTIMATE")
     (terpri)(terpri)(terpri)
     (princ "   Water                          ")
     (format t " 10D" (round (supply-item-water sup1)))
     (princ "   gallons")
     (terpri)
     (princ "   B-Rations                      ")
     (format t " 10D"  (supply-item-b-rations sup1))
     (princ "   rations")
     (terpri)
     (princ "   MRE-Ration                     ")
     (format t " 10D" (supply-item-mre-rations sup1))
     (princ "   rations ")
     (terpri)
     (princ "   Class II Supplies              ")
     (format t " 10D" (round (supply-item-class-II-supplies sup1)))
     (princ "   STONS ")
     (terpri)
     (princ "   Diesel Fuel                    ")
     (format t " 10D" (round (supply-item-diesel-fuel sup1)))
     (princ "   gallons ")
     (terpri)
     (princ "   Class IV Supplies              ")
     (format t " 10D" (round (supply-item-class-IV-supplies sup1)))
     (princ "   STONS ")
     (terpri)
     (princ "   Tank ammo 105mm                ")
     (format t " 10D"(supply-item-tank-ammo-105mm sup1))
     (princ "   rounds ")
     (terpri)
     (princ "   TOW ammo                       ")
     (format t " 10D" (supply-item-tow-ammo sup1))
     (princ "   rounds ")
     (terpri)
     (princ "   DRAGON ammo                    ")
     (format t " 10D" (supply-item-dragon-ammo sup1))
     (princ "   rounds ")
     (terpri)
     (princ "   Howitzer ammo 105mm            ")
     (format t " 10D" (supply-item-howitzer-ammo-105mm sup1))
     (princ "   rounds ")
     (terpri)
     (princ "   Howitzer ammo 155mm            ")
     (format t " 10D" (supply-item-howitzer-ammo-155mm sup1))
     (princ "   rounds ")
     (terpri)
     (princ "   Howitzer ammo 8 inch           ")
     (format t " 10D" (supply-item-howitzer-ammo-8in sup1))
     (princ "   rounds ")
     (terpri)
     (princ "   Vulcan ammo 20mm               ")
     (format t " 10D" (supply-item-vulcan-ammo-20mm sup1))
     (princ "   rounds")
     (terpri)
     (princ "   Mortar ammo 81mm               ")
     (format t " 10D" (supply-item-mortar-ammo-81mm sup1))
     (princ "   rounds ")
     (terpri)
     (princ "   Mortar ammo 107mm              ")
     (format t " 10D" (supply-item-mortar-ammo-107mm sup1))
     (princ "   rounds ")
     (terpri)
     (princ "   MG ammo .50 caliber            ")
     (format t " 10D" (supply-item-mg-ammo-.50-caliber sup1))
     (princ "   rounds ")
     (terpri)
```

```
(princ "   MG ammo 7.62mm              ")
(format t " 10D" (supply-item-mg-ammo-7.62mm sup1))
(princ "   rounds ")
(terpri)
(princ "   rifle ammo 5.56mm          ")
(format t " 10D" (supply-item-rifle-ammo-5.56mm sup1))
(princ "   rounds ")
(terpri)
(princ "   Class VII supplies         ")
(format t " 10D" (round (supply-item-class-VII-supplies sup1)))
(princ "   STONS ")
(terpri)
(princ "   Class VIII supplies        ")
(format t " 10D" (round (supply-item-class-VIII-supplies sup1)))
(princ "   STONS ")
(terpri)
(princ "   Class IX supplies          ")
(format t " 10D"(round (supply-item-class-IX-supplies sup1)))
(princ "   STONS ")
(terpri))
```

LIST OF REFERENCES

1. Department of the Army Pamphlet 350-31, *TMACS User's Manual-Battalion Level*, Fort McPhearson, Georgia, 1987.

2. Ivanov, Dmitrii Afanasevich, V.P. Savelyev, and P.V. Shemansky, *Fundamentals of Tactical Command and Control : a Soviet view*, translated and published under the auspices of the United States Air Force; published with the approval of the ALL-Union Copyright Agency of the U.S.S.R., Washington, D.C., U.S. Air Force, 1977.

3. Department of the Army Field Manual FM 101-10-1, *Staff Officers Field Manual, Organizational, Technical and Logistics Data*, Baltimore, Maryland, 1978.

4. U.S. Army Command and General Staff College, Student Text 101-2, *Planning Factors*, 1985.

5. Smith, Bernard T., *Focus Forecasting Computer Techniques for Inventory Control*, C.B.I. Publishing Company, Inc., 1978.

6. Foulds, L. R., *Combinatorial Optimization for Undergraduates*, Springer-Verlag, 1984.

7. Charniak, Eugene and Drew McDermott, *Introduction to Artificial Intelligence*, Addison-Wesley Publishing Company, 1986.

8. Millward, Richard B. and Thomas D. Wickens, "Concept-Identification Models" in : Krantz, D. H., et al., *Learning, Memory, and Thinking*, in volume I of the Contemporary Developments in Mathematical Psychology series, W. H. Freeman and Co., 1974.

9. Koffman, Elliott B., *Problem Solving and Structured Programming in Pascal*, Addison-Wesley Publishing Company, Inc., 1985.

10. Wilensky Robert, *Common Lispcraft*, W. W. Norton and Company, 1986.

11. MacLennan, Bruce J., *Principles of Programming Languages*, Holt, Rinehart and Winston, 1983.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Computer Technology Programs, Code 37 Naval Postgraduate School Monterey, CA 93943-5000	1
5.	Associate Professor Neil C. Rowe Code 52Rp Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5000	1
6.	Professor Robert B. McGhee Code 52Mz Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5000	1
7.	Commander United States Army Logistics Center Fort Lee, VA 23801-6000	1
8.	The Quartermaster General, United States Army Fort Lee, VA 23801-5032	1
9.	Defense Logistics Studies Information Exchange U.S. Army Logistics Management Center Fort Lee, VA 23801-6043	1
10.	Mr. Russell Davis HQ, USACDEC ATTN : ATEC-IM Fort Ord, CA 92152	1

11. Captain Mark J. Davis 3
80 Moore Lane
Washingtonville, NY 10992

12. Mr.& Mrs. Joseph W. Davis 1
3321 Spiceland Drive
Boise, ID 83704